Tensorized Self-Attention: Efficiently Modeling Pairwise and Global Dependencies Together

Tao Shen[1], **Tianyi Zhou**[2], **Guodong Long**[1], **Jing Jiang**[1], **Chengqi Zhang**[1]
[1] Centre for Artificial Intelligence, School of Software, University of Technology Sydney
[2] Paul G. Allen School of Computer Science & Engineering, University of Washington
tao.shen@student.uts.edu.au, tianyizh@uw.edu,
{guodong.long, jing.jiang, chengqi.zhang}@uts.edu.au

Abstract

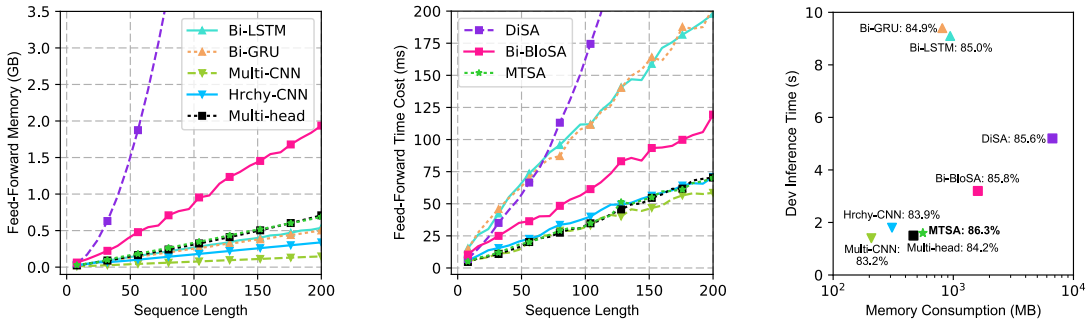
Neural networks equipped with self-attention have parallelizable computation, light-weight structure, and the ability to capture both long-range and local dependencies. Further, their expressive power and performance can be boosted by using a vector to measure pairwise dependency, but this requires to expand the alignment matrix to a tensor, which results in memory and computation bottlenecks. In this paper, we propose a novel attention mechanism called "Multi-mask Tensorized Self-Attention" (MTSA), which is as fast and as memory-efficient as a CNN, but significantly outperforms previous CNN-/RNN-/attention-based models. MTSA 1) captures both pairwise (token2token) and global (source2token) dependencies by a novel compatibility function composed of dot-product and additive attentions, 2) uses a tensor to represent the feature-wise alignment scores for better expressive power but only requires parallelizable matrix multiplications, and 3) combines multi-head with multi-dimensional attentions, and applies a distinct positional mask to each head (subspace), so the memory and computation can be distributed to multiple heads, each with sequential information encoded independently. The experiments show that a CNN/RNN-free model based on MTSA achieves state-of-the-art or competitive performance on nine NLP benchmarks with compelling memory- and time-efficiency.

1 Introduction

Recurrent neural network (RNN) and convolutional neural network (CNN) have been broadly used as context fusion modules for natural language processing (NLP) tasks. Recently, RNN/CNN in conjunction with an attention mechanism has been proven to be effective for contextual feature modeling in a wide range of NLP tasks, including sentiment classification (Li et al., 2018), machine translation (Bahdanau et al., 2015), reading comprehension (Seo et al., 2017; Yu et al., 2018), etc. More recently, self-attention mechanisms have been developed for context fusion and syntactic dependency modeling with the advantage of fewer parameters, more parallelizable computation, and better empirical performance (Hu et al., 2017; Vaswani et al., 2017; Shen et al., 2018a). In addition, neural networks based solely on self-attention mechanisms have achieved state-of-the-art quality on many NLP tasks, e.g., machine translation (Vaswani et al., 2017), sentence embedding (Shen et al., 2018a) and semantic role labeling (Tan et al., 2017).

Self-attention mechanisms can be categorized into two classes according to the type of dependency each aims to model. The first category is *token2token* self-attention (Hu et al., 2017; Vaswani et al., 2017; Shen et al., 2018a) that captures syntactic dependency between every two tokens in a sequence. An efficient dot-product compatibility function is usually deployed to measure this pairwise dependency (Vaswani et al., 2017). In contrast, additive compatibility function captures the dependency by multi-layer perceptron (MLP), and can usually achieve better performance (Britz et al., 2017). Its expressive power can be further improved if expanded to multiple dimensions (Shen et al., 2018a). This multi-dim self-attention empirically surpasses dot-product one, but suffers from expensive computation and memory, which grow linearly with the number of features and quadratically with the sequence length. Hence, it is not scalable to long sequences in practice.

The second category is *source2token* self-attention (Liu et al., 2016; Lin et al., 2017; Shen et al., 2018a) aiming to capture global dependency, i.e., the importance of each token to the entire sequence for a specific task. Its time and space complexities grow linearly, rather than quadratically,



(a) (b) (c)

Figure 1: (a) Memory consumption and (b) time cost *vs.* sequence length on synthetic data; (c) memory load (x-axis), inference time on dev set (y-axis) and test accuracy on the SNLI dataset.

with the sequence length. Hence, it is empirically efficient in terms of memory and computation even if expanded to multiple dimensions, i.e., using a vector of feature-wise scores instead of a scalar for the global dependency. But, it is hard to reach state-of-the-art performance on NLP tasks due to the lack of pairwise and local dependencies.

In this paper, we propose a novel attention mechanism called multi-mask tensorized self-attention (MTSA), for context fusion. In MTSA, 1) the pairwise dependency is captured by an efficient dot-product based token2token self-attention, while the global dependency is modeled by a feature-wise multi-dim source2token self-attention, so they can work jointly to encode rich contextual features; 2) self-attention alignment scores are tensorized for more expressive power in that each pair of tokens has one score for each feature, but no tensor computation is required other than simple and efficient matrix multiplications when implemented; 3) the tensors above are computed in multiple subspaces (i.e., in a multi-head fashion) rather than in the original input space, so the required memory and computation can be distributed to multiple subspaces; and 4) a distinct positional mask is applied to each head in order to encode rich structural information such as the sequential order and relative position of tokens.

In the experiments, we build CNN/RNN-free neural networks based on MTSA for sentence embedding and sequence tagging tasks, including natural language inference, semantic role labeling, sentiment analysis, question-type classification, machine translation, etc. The results demonstrate that MTSA achieves state-of-the-art or competitive performance on nine benchmark datasets. To summarize the comparison of MTSA with recently popular models, we show the memory consumption and time cost *vs.* sequence length respectively in Figure 1(a) and 1(b) on synthetic data (batch size of 64 and feature channels of 300). On the SNLI (Bowman et al., 2015), a public dataset for language inference, as shown in Figure 1(c), MTSA achieves the best result but is as fast and as memory-efficient as the CNNs (all baselines and the benchmark are detailed in Section 4).

Notations: 1) lowercase denotes a vector; 2) bold lowercase denotes a sequence of vectors (stored as a matrix); and 3) uppercase denotes a matrix or tensor.

2 Background

2.1 Attention Mechanism

Given an input sequence of token embeddings or memory slots $\boldsymbol{x} = [x_1, \ldots, x_n] \in \mathbb{R}^{d_e \times n}$, and a vector representation of a query $q \in \mathbb{R}^{d_q}$, attention mechanism (Bahdanau et al., 2015; Luong et al., 2015) computes an alignment score between each token x_i and q by a compatibility function $f(x_i, q)$, which aims to measure the dependency/relevance between x_i and q, or the attention of q to x_i, w.r.t. a given task. The scores are transformed into probabilities through a $\mathrm{softmax}$ function. These probabilities are then used as weights to sum all the tokens and generate a contextual embedding for q, i.e.,

$$p(z|\boldsymbol{x}, q) = \mathrm{softmax}(a), \quad a = [f(x_i, q)]_{i=1}^{n},$$
$$s = \sum_{i=1}^{n} p(z = i|\boldsymbol{x}, q) \cdot x_i = \mathbb{E}_{i \sim p(z|\boldsymbol{x}, q)}[x_i], \quad (1)$$

where $a \in \mathbb{R}^n$ denotes the vector of n alignment scores, $p(z|\boldsymbol{x}, q)$ is the categorical distribution for

attention probabilities, which is derived from applying $\mathrm{softmax}$ function to a. And, $s \in \mathbb{R}^{d_e}$ is the output vector for the query q.

There are two major types of compatibility functions, leading to the two most frequently used attention mechanisms. The first one is dot-product or multiplicative compatibility function (Eq.(2)), which composes **dot-product attention** mechanism (Luong et al., 2015) using cosine similarity to model the dependencies. The other one is additive or multi-layer perceptron (MLP) compatibility function (Eq.(3)) that results in **additive attention** mechanism (Bahdanau et al., 2015) using MLP to model the dependencies.

$$f(x_i, q) = \langle W^{(d1)} x_i, W^{(d2)} q \rangle, \quad (2)$$

$$f(x_i, q) = w^T \sigma_a(W^{(a)}[x_i; q] + b^{(a)}) + b, \quad (3)$$

where $W^{(d1)} \in \mathbb{R}^{d_i \times d_e}, W^{(d2)} \in \mathbb{R}^{d_i \times d_q}, W^{(a)} \in \mathbb{R}^{d_a \times (d_e + d_q)}, w \in \mathbb{R}^{d_a}$ are learnable parameters, $\langle \cdot, \cdot \rangle$ denotes inner-product. Empirically, networks with additive attention usually outperform those with dot-product attention, but require more computation time and memory (Britz et al., 2017).

Multi-dim attention mechanism (Shen et al., 2018a) expands the alignment score in previous attention mechanisms to a vector for feature-wise scores, each computed on a feature dimension. It has greater capacity to model complex dependencies, and can handle context variation and polysemy problems harassing many NLP tasks. **In particular, it replaces vector $w^T \in \mathbb{R}^{1 \times d_a}$ in additive compatibility function (Eq.(3)) with a matrix $W \in \mathbb{R}^{d_e \times d_a}$, and thus produces d_e scores to describe the attention of q to x_i.**

2.2 Self-Attention Mechanism

Self-attention mechanism is a special case of attention mechanisms, where the query q stems from the input sequence itself. Self-attention mechanisms can be classified into token2token or source2token self-attention mechanism according to the type of dependency each aims to model.

A) Token2token self-attention mechanism (Vaswani et al., 2017; Shen et al., 2018a) aims at producing a context-aware representation for each token in light of its syntactic dependencies on other tokens from the same sequence. Two examples of token2token self-attention are 1) scaled dot-product self-attention which composes the multi-head self-attention (Vaswani et al., 2017), and 2) masked self-attention used in directional self-attention (Shen et al., 2018a).

A.1) Scaled dot-product attention mechanism (Vaswani et al., 2017) in general form has three arguments: query tokens $\boldsymbol{q} \in \mathbb{R}^{d_i \times m}$, key tokens $\boldsymbol{k} \in \mathbb{R}^{d_i \times n}$ and value tokens $\boldsymbol{v} \in \mathbb{R}^{d_h \times n}$ associated with the key tokens. It uses a scaled dot-product function to model the relationship between each query and key, and finally outputs a sequence $\boldsymbol{s} = [s_1, \ldots, s_m] \in \mathbb{R}^{d_h \times m}$ such that

$$\boldsymbol{s} = \mathrm{sdpAttn}(\boldsymbol{q}, \boldsymbol{k}, \boldsymbol{v}) \triangleq \boldsymbol{v}\, \mathrm{softmax}(\frac{\boldsymbol{q}^T \boldsymbol{k}}{\sqrt{d_q}})^T \quad (4)$$

A special case of this mechanism is that the three input arguments are derived from the same source, i.e., $\boldsymbol{q}/\boldsymbol{k}/\boldsymbol{v} = f^{q/k/v}(\boldsymbol{x})$, which can be referred to as a token2token self-attention, namely scaled dot-product self-attention. As for **multi-head attention** mechanism, the input is projected into multiple subspaces, then parameter-untied scaled dot-product attention is applied to the embeddings in each subspace. The results for multiple subspaces are concatenated to form the final output $\boldsymbol{s}$, i.e.,

$$\boldsymbol{s} = W^{(o)}[H_1; \ldots; H_h], \quad (5)$$
$$\text{where } H_c = \mathrm{sdpAttn}(W_c^q \boldsymbol{q}, W_c^k \boldsymbol{k}, W_c^v \boldsymbol{v}).$$

A.2) Masked self-attention mechanism (Shen et al., 2018a) uses multi-dim compatibility function to model the dependency between every two tokens in a sequence, and uses positional mask to encode sequential information. It overcomes inherent problem appearing in self-attention compared to RNNs on the lack of sequential information. Its compatibility function is defined as

$$f(x_i, x_j) = c \cdot \tanh\{(W^{(m)}[x_i; x_j] + b^{(m)})/c\} + M_{i,j} \quad (6)$$

where c is a constant scalar, $W^{(m)} \in \mathbb{R}^{d_e \times 2d_e}$ is learnable weight matrix, and M is a positional mask with each entry $M_{i,j} \in \{-\infty, 0\}$. When $M_{i,j} = -\infty$, applying $\mathrm{softmax}$ function to the alignment scores results in a zero attention probability, which cuts off the attention of x_j to x_i. Hence, masked self-attention with an asymmetric mask, where $M_{ij} \neq M_{ji}$, can encode sequential or other structural information (Shen et al., 2018a; Im and Cho, 2017). To this end, two positional masks have been proposed to encode the forward and backward order information respectively, i.e.,

$$M_{i,j}^{fw} = \begin{cases} 0, & i < j \\ -\infty, & \text{otherwise} \end{cases} \quad M_{i,j}^{bw} = \begin{cases} 0, & i > j \\ -\infty, & \text{otherwise} \end{cases}$$

Furthermore, **directional self-attention** (DiSA) (Shen et al., 2018a) concatenates the features produced by masked self-attention mechanisms with the forward and backward positional masks (i.e., M^{fw}, M^{bw}), leading to context-ware representations with bi-directional information encoded.

B) Source2token self-attention mechanism (Liu et al., 2016; Lin et al., 2017; Shen et al., 2018a) is designed for sentence embedding or sequence compression, which is based on the importance of each token x_i to the entire source sequence $\boldsymbol{x}$ for a specific task. Specifically, it removes the query q from the compatibility function $f(x_i, q)$ when computing the alignment score. For example, the compatibility function of additive source2token self-attention mechanism is to simply remove q from Eq.(3).

3 Proposed Models

In this section, we firstly elaborate on tensorized self-attention (TSA) in Section 3.1, which captures both pairwise and global dependencies by combining the two types of self-attention mechanisms introduced in Section 2.2. Then, we extend TSA to multi-mask tensorized self-attention (MTSA) in Section 3.2 by applying different positional masks to TSA in multiple subspaces (multi-head fashion). Lastly, in Section 3.3, we present an efficient computation scheme for MTSA without any high-rank tensor computation involved even if tensorized alignment scores are used.

3.1 Tensorized Self-Attention (TSA)

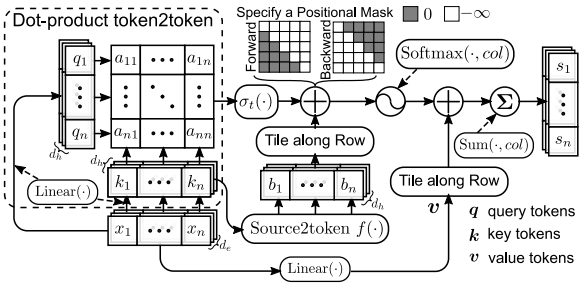


Figure 2: Tensorized self-attention (TSA) Mechanism.

Tensorized self-attention (TSA), whose structure is illustrated in Figure 2, is a neural mechanism that can be trained to model both pairwise and global dependencies, while any previous self-attention mechanism only focuses on one type of dependencies. TSA models both types by combining the aforementioned token2token and source2token self-attention mechanisms. This generates an $n \times n \times d_h$ tensor containing the alignment scores between every two tokens on each feature dimension. These scores are then normalized and transformed into probability weights, which are used to sum all dependent tokens and then generate the contextual embedding for each input token. We will demonstrate later in Section 3.3 that only matrix rather than tensor operation is required when executing the procedures above.

To facilitate the elaboration of proposed models and keep the consistent notation with prior attention mechanisms, TSA first projects the input embeddings $\boldsymbol{x}$ into three spaces to represent the query, key and value tokens, respectively.

$$\boldsymbol{q} = W^{(t1)}\boldsymbol{x}, \ \boldsymbol{k} = W^{(t2)}\boldsymbol{x}, \text{ and } \ \boldsymbol{v} = W^{(t3)}\boldsymbol{x}, \quad (7)$$

where $W^{(t1)}, W^{(t2)} \in \mathbb{R}^{d_i \times d_e}$ and $W^{(t3)} \in \mathbb{R}^{d_h \times d_e}$ are learnable weights for projections.

TSA then integrates two kinds of compatibility functions from two self-attention mechanisms respectively. Firstly, the scaled dot-product self-attention is used to capture dependency between every two tokens. Dot-product operations are fast, and sufficient to model the **pairwise dependency** in most tasks. Its compatibility function is

$$f^t(k_i, q_j) = \langle k_i, \ q_j \rangle / \sqrt{d_i}, \ \ \forall i, j \in [n], \quad (8)$$

where $\langle \cdot, \cdot \rangle$ is inner-product operation. Then, a multi-dim source2token self-attention mechanism is used to estimate the contribution of each token to the given task on each feature dimension. It aims at capturing the importance of each token to the entire input sequence w.r.t. the task, i.e., the **global dependency**. The multi-dim extension only linearly increases the memory and computation of source2token self-attention by a multiplicative factor d_h, but is essentially helpful to improve expressive capability in line with prior works (Shen et al., 2018a). Its compatibility function is

$$f^s(k_i) = W^{(s2)}\sigma_m(W^{(s1)}k_i + b^{(s1)}) + b^{(s2)}, \quad (9)$$

where $\forall i \in [n]$, $W^{(s1)} \in \mathbb{R}^{d_a \times d_i}, W^{(s2)} \in \mathbb{R}^{d_h \times d_a}$ are the learnable weights, and $\sigma_m(\cdot)$ is an activation function. The compatibility function used in TSA broadcasts the scalar alignment score $f^t(k_i, q_j) \in \mathbb{R}$ computed by the token2token self-attention to all d_h feature dimensions, and then adds them to the feature-wise score vector

$f^s(k_i) \in \mathbb{R}^{d_h}$ computed by the source2token self-attention. In addition, the positional masks from masked self-attention (in Section 2.2) are also integrated to encode sequential and structural information. These yield following compatibility function of TSA.

$$\begin{aligned}\left[f^{tsa}(k_i, q_j)\right]_l = \\ \sigma_t\left(f^t(k_i, q_j)\right) + \sigma_s([f^s(k_i)]_l) + M_{i,j},\end{aligned} \tag{10}$$

where $\forall i, j \in [n]$, $\forall l \in [d_h]$. $\sigma_t(\cdot)$ and $\sigma_t(\cdot)$ are two scale functions. They control the way to combine two kinds of scores and their weights.

For each query token q_j, a $\mathrm{softmax}$ function is applied to the alignment scores $[f^{tsa}(k_i, q_j)]_{i=1}^n$ on each feature dimension, resulting in a categorical distribution over all value tokens $[v_i]_{i=1}^n$ based on corresponding key tokens $[k_i]_{i=1}^n$. The probability of token q_j attending to v_i on the l^{th} feature dimension (i.e., $z_l = i$) is

$$p(z_l = i|\boldsymbol{k}, q_j) \triangleq [p_i^j]_l \triangleq \frac{e^{\left[f^{tsa}(k_i, q_j)\right]_l}}{\sum_{g=1}^n e^{[f^{tsa}(k_g, q_j)]_l}}, \tag{11}$$

where, $\forall i, j \in [n]$, $\forall l \in [d_h]$. TSA outputs a contextual embedding for each input token on every feature dimension as the weighted sum of all the value token embeddings on that dimension, where the weights are provided by the probabilities in Eq.(11). It is the expectation of sampling a value token embeddings on each feature dimension according to the feature-wise probability distribution, i.e.,

$$\begin{aligned}&\boldsymbol{s} \triangleq [s_j]_{j=1}^n, \quad \text{where} \\ &s_j \triangleq \left[\mathbb{E}_{i\sim p(z_l|\boldsymbol{k}, q_j)}([v_i]_l)\right]_{l=1}^{d_h} = \sum\nolimits_{i=1}^n p_i^j \cdot v_i\end{aligned} \tag{12}$$

3.2 Multi-Mask Tensorized Self-Attention (MTSA) Mechanism

Rather than computing attention in the original input space, multi-head attention (Vaswani et al., 2017) projects the input sequence to multiple subspaces, applies attention to the projected embedding in each subspace, and concatenates their outputs at last. The computations associated with multiple heads can be completed in parallel. By using adequate heads, each with a low-dimensional subspace (i.e., the representation dimension for each head is updated by $d_h \leftarrow d_h/h$ where h is the number of head), it reduces parameters and memory/computation cost and increases diversity of the attention. In addition, to encode different kinds of sequential or structural information, multiple different positional masks (e.g., forward, backward and multi-length window) can be further applied to the multiple heads.

The memory-/time-efficiency and expressive power of TSA can be improved by using the combination of the multi-head and multi-mask techniques introduced above. By writing TSA mechanism as a function $\mathrm{TSA}(\boldsymbol{x}, M)$ with input sequence $\boldsymbol{x} \in \mathbb{R}^{d_e \times n}$ and a positional mask $M \in \mathbb{R}^{n\times n}$, and the output given by Eq.(12), multi-mask tensorized self-attention (MTSA) produces

$$\begin{aligned}\boldsymbol{s} =& W^{(o)}[H_1; \ldots; H_h], \\ &\text{where } H_c = \mathrm{TSA}^c(\boldsymbol{x}, M^c),\end{aligned} \tag{13}$$

where $W^{(o)} \in \mathbb{R}^{h\cdot d_h \times h\cdot d_h}$, h is the number of heads, TSA^c denotes the c^{th} parameter-independent TSA block that produces a d_h-dim representation in the c^{th} subspace, M^c represents the positional mask applied to attention in the c^{th} subspace, $[\cdot; \ldots; \cdot]$ denotes a vertical concatenation operation, and $\boldsymbol{s} \in \mathbb{R}^{h\cdot d_h \times n}$ is the output of MTSA. In our experiments, we apply forward mask to half of the heads and apply backward mask to the other half to encode bi-directional order information of the input sequence.

3.3 Computation-Optimized MTSA

Algorithm 1 Multi-Mask Tensorized Self-Attention

Input: input sequence $\boldsymbol{x} \in \mathbb{R}^{d_e \times n}$, head number h, subspace dimension d_h, positional masks $\{M^c\}_{c=1}^h$, and weights/biases: $\{W_c^{(t1)}, W_c^{(t2)} \in \mathbb{R}^{d_i \times d_e}, W_c^{(t3)} \in \mathbb{R}^{d_h \times d_e}, W_c^{(s1)} \in \mathbb{R}^{d_a \times d_i}, W_c^{(s2)} \in \mathbb{R}^{d_h \times d_a}$, and $b_c^{(s1)}$, $b_c^{(s2)}\}_{c=1}^h$, and $W^{(o)}$

Output: contextual embeddings $\boldsymbol{s} = [s_1, \ldots, s_n] \in \mathbb{R}^{h\cdot d_h \times n}$

for all $c = 1, \ldots, h$ **do** ▷ Computing h-head in parallel
  $\boldsymbol{q^c}$, $\boldsymbol{k^c}$, $\boldsymbol{v^c} \leftarrow W_c^{(t1)}\boldsymbol{x}$, $W_c^{(t2)}\boldsymbol{x}$, $W_c^{(t3)}\boldsymbol{x}$
  $R_c \leftarrow \frac{(\boldsymbol{k^c})^T\boldsymbol{q^c}}{\sqrt{d_h}}$ ▷ $n \times n$ token2token attention scores
  $S_c \leftarrow W_c^{(s2)}\sigma_m(W_c^{(s1)}\boldsymbol{k^c} + b_c^{(s1)}) + b_c^{(s2)}$
    ▷ $d_h \times n$ scores of source2token attention
  $E_c^R \leftarrow \exp(\sigma_t(R_c))\cdot\exp(M^c)$
    ▷ Applying mask M^c to token2token weights
  $E_c^S \leftarrow \exp(\sigma_s(S_c))$; $E_c^X \leftarrow \boldsymbol{v^c}\cdot E_c^S$
    ▷ Applying source2token weights E_c^S to $\boldsymbol{v^c}$
  $H_c \leftarrow E_c^X E_c^R / E_c^S E_c^R$ ▷ Applying masked token2token weights E_c^R and normalizing
end for
Return $\boldsymbol{s} \leftarrow W^{(o)}[H_1; \ldots; H_h]$
  ▷ Vertical concatenation of the outputs from all h heads

As shown in Eq.(10) and Eq.(11), TSA or each head of MTSA needs to compute the attention

scores and probabilities as $n \times n \times d_h$ tensors. In accordance with multi-dim self-attention (Shen et al., 2018a), this makes TSA more expressively powerful and improves the final performance for sequence modeling, but terribly leads to memory explosion and computational bottleneck on long sequences with large n and d_h. Fortunately, in MTSA, it is possible to significantly reduce the demand on computations to matrix-only operations by exploring the computational structure.

A memory-optimized and highly-parallelizable computation scheme for MTSA is given in Algorithm 1. For each head, the score matrices of token2token and source2token are computed in steps 3 and 4 respectively. Then, we combine token2token scores with the positional mask to form a new mask in step 5, and compute the $d_h \times n$ output embedding with the weighs from the multi-dim source2token self-attention in step 6. Finally, in step 7, we apply the new mask from step 5 to the weighted embedding from step 6 and complete the normalization. This procedure generates the exactly same output as Eq.(13)but no any tensor operation is incurred.

4 Experiments

We compare MTSA with commonly-used context fusion baselines on several NLP tasks[1]. When addressing a sentence embedding problem, a multi-dim source2token self-attention is applied on the top of context fusion module to produce the sequence embedding. Codes are implemented in Python with Tensorflow and executed on a single NVIDIA GTX 1080Ti graphics card. In addition, data for both time cost and memory consumption are collected under Tensorflow-1.7 with CUDA9 and cuDNN7.

The context fusion baselines include **1) Bi-LSTM** (Graves et al., 2013): 600D bi-directional LSTM consisting of 300D forward plus 300D backward LSTMs, **2) Bi-GRU** (Chung et al., 2014): 600D bi-directional GRU, **3) Multi-CNN** (Kim, 2014): three CNNs with 200D kernels to model 3/4/5-grams respectively, **4) Hrchy-CNN** (Gehring et al., 2017): 3-layer 300D stacked CNN with kernel size 5, gated linear units (Dauphin et al., 2016) and residual connections (He et al., 2016), **5) Multi-head** (Vaswani et al., 2017): 600D multi-head self-attention with 8 heads (75-dim subspace per head) and positional embedding used by Vaswani et al. (2017), **6) DiSA** (Shen et al., 2018a): 600D directional self-attention mechanism consisting of 300D forward and 300D backward masked self-attentions, and **7) Bi-BloSA** (Shen et al., 2018c): 600D bi-directional block self-attention with intra-/inter-block self-attention, aiming to reduce the time and space complexities of multi-dim self-attention by using hierarchical structure.

4.1 Natural Language Inference

Natural language inference (NLI) aims at speculating on the relationship between a premise and a corresponding hypothesis, where the relationship could be *entailment*, *neutral* or *contradiction*. In experiments, we first compare MTSA with other baselines on the Stanford Natural Language Inference (Bowman et al., 2015) (SNLI) dataset.

Following the method of applying *sentence-encoding* model to NLI given by Bowman et al. (2016), two parameter-tied sentence-encoding models are used to generate embeddings for premise and hypothesis, resulting in s^p and s^h respectively. The concatenation of s^p, s^h, $s^p - s^h$ and $s^p \odot s^h$ representing the relationship is passed into a 3-way neural classifier for final prediction.

The experimental results of the models from the official leaderboard, baselines, and MTSA are shown in Table 1. MTSA achieves state-of-the-art performance with less time and memory cost. Compared to the methods from the leaderboard, MTSA outperforms RNN-based encoders (e.g., Residual stacked enc.), RNN+attention encoders (e.g., Deep Gated Attn.) and even parsing trees based encoders (e.g., Gumbel TreeLSTM enc.) by a large margin. Compared to the two competitive self-attention networks with complicated and expensive training computations, MTSA trained in end-to-end manner achieves the same state-of-the-art performance by using much fewer parameters and less computational time.

Compared to baselines, MTSA is $4 \sim 5\times$ faster than RNN-based models and outperforms CNN-based models given a similar number of parameters and computation time. Moreover, compared to the dot-product self-attention (Multi-head), MTSA costs similar time and memory but performs more expressively powerful self-attention, and thus achieves better performance. Furthermore, compared to the multi-dim self-

[1]Codes for Experiments are released at https://github.com/taoshen58/mtsa.

Model	$\|\theta\|$	Time/Epoch	Inf. Time	Memory	Train Acc.	Test Acc.
300D SPINN-PI encoders (Bowman et al., 2016)	3.7m				89.2	83.2
600D Bi-LSTM encoders (Liu et al., 2016)	2.0m				86.4	83.3
600D Bi-LSTM enc.+intra-attn (Liu et al., 2016)	2.8m				84.5	84.2
600D Deep Gated Attn. (Chen et al., 2017)	11.6m				90.5	85.5
600D Gumbel TreeLSTM enc. (Choi et al., 2018)	10.0m				93.1	86.0
600D Residual stacked enc. (Nie and Bansal, 2017)	29.0m				91.0	86.0
300D Reinforced SAN (Shen et al., 2018b)	3.1m	404s			92.6	86.3
Distance-based SAN (Im and Cho, 2017)	4.7m	416s			89.6	86.3
Bi-LSTM (Graves et al., 2013)	2.9m	854s	9.1s	942MB	90.4	85.0
Bi-GRU (Chung et al., 2014)	2.5m	850s	9.4s	810MB	91.9	84.9
Multi-CNN (Kim, 2014)	1.4m	137s	1.4s	208MB	89.3	83.2
Hrchy-CNN (Gehring et al., 2017)	3.4m	195s	1.8s	309MB	91.3	83.9
Multi-head (Vaswani et al., 2017)	2.0m	179s	1.5s	466MB	89.6	84.2
DiSA (Shen et al., 2018a)	2.3m	390s	5.2s	6682MB	91.1	85.6
Bi-BloSA (Shen et al., 2018c)	4.1m	303s	3.2s	1600MB	91.6	85.8
MTSA	2.9m	180s	1.6s	558MB	91.8	**86.3**

Table 1: Experimental results for different methods with comparative parameter number on SNLI. $|\theta|$: the number of parameters (excluding word embedding part); Time/Epoch: averaged training time per epoch with batch size 128; Inf. Time: averaged dev inference time with batch size 128; Memory: memory load on synthetic data of sequence length 64 and batch size 64 with back-propagation considered; Train Acc. and Test Acc.: the accuracies on training/test sets. All state-of-the-art methods in leaderboard are listed in Table 1&2 up to Sep. 2018.

Model	SNLI		MultiNLI	
	Dev	Test	Match	Mismatch
BiLSTM w/ Shortcut[a]	–	86.0	74.6	73.6
BiLSTM w/ Gen-Pooling[b]	–	86.6	73.8	74.0
HBMP[c]	–	86.6	73.7	73.0
Transfer + Multi-Head	86.9	86.6	76.3	75.7
Transfer + MTSA	**87.2**	**86.9**	**76.7**	**76.4**

Table 2: Experimental results on *sentence-encoding* based SNLI and MultiNLI benchmark tasks. "*Transfer*" denotes pretrained language model on large corpus for transfer learning, which detailed by Radford et al. (2018). References: [a](Nie and Bansal, 2017), [b](Chen et al., 2018), [c](Talman et al., 2018).

Model	$\|\theta\|$	Inf. Time	Test Acc.
MTSA	2.9m	1.6	86.3
MTSA w/o fw&bw masks	2.9m	1.6	85.3 (-1.0)
MTSA w/o token2token	2.5m	1.5	85.8 (-0.5)
MTSA w/o source2token	2.5m	1.4	84.9 (-1.4)
MTSA w/o proposed modules	1.8m	1.1	84.3 (-2.0)

Table 3: An ablation study of MTSA on SNLI.

attention (DiSA and Bi-BloSA), MTSA uses much less memory and time but even produces much better prediction quality.

In addition, to further improve the state-of-the-art performance, in contrast to training from scratch, a language model built on the Transformer (Vaswani et al., 2017) unsupervisedly pretrained on large English corpus (detailed by Radford et al. (2018)) is transfered for the baseline and proposed models for *sentence-encoding* based NLI tasks. As shown in Table 2, MTSA integrated with pretrained language model can achieve new state-of-the-art accuracy on both SNLI and Multi-Genre Natural Language Inference (MultiNLI) (Williams et al., 2017)[2] among all *sentence-encoding* models.

An ablation study of MTSA is shown in Table 3 to verify the capability of its each part in context fusion. The results show that token2token (modeling pairwise dependency), source2token (modeling global dependency), and positional masks (encoding sequential information) all contribute important information to sequence modeling, and the contributions are complementary.

4.2 Semantic Role Labeling

To verify the capability of MTSA in generating context-aware representation of each token, we compare it with baselines on semantic role labeling (SRL) task, which aims to tag each token from an input sequence with a label for its semantic role. Particularly, given a sentence, the goal of SRL is to identify the arguments of each target verb into semantic roles, which can benefit many downstream NLP tasks. SRL has two steps:

[2]All test results are Evaluated on Kaggle official websites: https://www.kaggle.com/c/multinli-matched-open-evaluation and https://www.kaggle.com/c/multinli-mismatched-open-evaluation

Models	Training Time	Development				WSJ Test				Brown Test			
		P	R	F1	Comp.	P	R	F1	Comp.	P	R	F1	Comp.
Täckström et al. (2015)		81.2	76.2	78.6	54.4	82.3	77.6	79.9	56.0	**74.3**	68.6	71.3	39.8
Zhou and Xu (2015)		79.7	79.4	79.6	-	82.9	82.8	82.8	-	70.7	68.2	69.4	-
He et al. (2017)		81.6	81.6	81.6	62.3	83.1	83.0	83.1	64.3	72.8	71.4	72.1	44.8
He et al. (2018)		-	-	-	-	-	-	83.9	-	-	-	73.7	-
Strubell et al. (2018)		-	-	-	-	**84.7**	84.2	84.5	-	73.9	72.4	73.1	-
Bi-LSTM (Graves et al., 2013)	72h	81.8	83.4	82.6	63.3	83.0	84.0	83.5	64.6	72.3	72.8	72.5	46.8
Multi-CNN (Kim, 2014)	19h	75.2	79.6	77.3	53.6	77.3	80.9	79.0	55.5	68.3	70.3	69.3	41.9
Multi-head* (Tan et al., 2017)	20h	82.6	83.6	83.1	65.2	84.5	85.2	**84.8**	66.4	73.5	**74.6**	74.1	48.4
MTSA	20h	**82.8**	**84.4**	**83.6**	**65.4**	84.2	**85.3**	**84.8**	**67.0**	**74.3**	**74.6**	**74.5**	**49.1**

Table 4: Experimental Results of SRL for single models on CoNLL-05 with gold predicates. *Multi-head baseline is equivalent to the model in Tan et al. (2017). For fair comparisons, first, we use the hyper-parameters provided by Tan et al. (2017) instead of tuning them; second, all listed models are independent of external linguistics information, e.g., PoS, dependency parsing.

Model	CR	MPQA	SUBJ	TREC	SST-5
cBoW[a]	79.9	86.4	91.3	87.3	/
Skip-thought[b]	81.3	87.5	93.6	92.2	/
DCNN[c]	/	/	/	93.0	48.5
SRU[d]	84.8(1.3)	89.7(1.1)	93.4(0.8)	93.9(0.6)	/
CNNs[d]	82.2(.2)	88.8(1.2)	92.9(0.7)	93.2(0.5)	/
Bi-LSTM	84.6(1.6)	90.2(0.9)	**94.7(0.7)**	94.4(0.3)	49.9(0.8)
Multi-head	82.6(1.9)	89.8(1.2)	94.0(0.8)	93.4(0.4)	48.2(0.6)
DiSA	84.8(2.0)	90.1(0.4)	94.2(0.6)	94.2(0.1)	51.0(0.7)
Bi-BloSA	84.8(0.9)	90.4(0.8)	94.5(0.5)	94.8(0.2)	50.6(0.5)
MTSA	**84.9(2.4)**	**90.5(0.6)**	94.5(0.6)	**95.3(0.3)**	**51.3(0.7)**

Table 5: Experimental results on five sentence classification benchmarks. References: [a](Mikolov et al., 2013), [b](Kiros et al., 2015), [c](Kalchbrenner et al., 2014), [d](Lei and Zhang, 2017).

1) assigning either a semantic argument or non-argument to a given predicate and 2) labeling a specific semantic role for the identified argument.

We follow the experimental setup in Tan et al. (2017), where the SRL task is treated as a BIO tagging problem. Tan et al. (2017) designed a deep attentive neural net by stacking multi-head self-attention, named as deepatt, to perform context fusion, whose output is then passed to a neural classifier to make the final decision. The results achieved by previous methods, baselines, and MTSA are shown in Table 4, which demonstrates that MTSA achieves new state-of-the-art performance on the CoNLL-05 dataset by costing similar training time as CNN and multi-head self-attention baselines.

4.3 Sentence Classifications

The goal of sentence classification is to predict the correct label for a sentence in various scenarios. We evaluate the models on five sentence classification benchmarks for different NLP tasks, which include **1) CR** (Hu and Liu, 2004): customer reviews of various products to predict whether the review is positive or negative, **2) MPQA** (Wiebe et al., 2005): an opinion polarity detection subtask of the MPQA dataset, **3) SUBJ** (Pang and Lee, 2004): subjectivity dataset where a label indicates whether a sentence is subjective or objective, **4) TREC** (Li and Roth, 2002): question-type classification dataset which classifies the question sentences into six classes, **5) SST-5** (Socher et al., 2013): the Stanford Sentiment Treebank dataset with five sentiment labels. The reported accuracies for CR, MPQA, and SUBJ are the mean of 10-fold cross validation. The accuracies for TREC are the mean of five runs on the dev set, and the accuracies for SST-5 are the mean of five runs on the test set. All standard deviations are shown in parentheses.

The prediction accuracies achieved on these five benchmarks are shown in Table 5. MTSA achieves the best prediction accuracy on CR, MPQA, TREC and SST-5 benchmarks with better time efficiency and a lower memory load.

4.4 Machine Translation

We also evaluate proposed model on WMT 2014 English-German translation task for exhaustive comparisons with multi-head attention. We replace multi-head self-attention modules in the encoder of official Transformer implementation with MTSA module and do not tune the hyperparameters. Although our computation resources is limited, we use two training setups and also introduce t-test to ensure that MTSA consistently outperforms multi-head self-attention in Transformer.

Model	Multi-head (Transformer)	MTSA
Param#	61.38M	61.58M
Setup1	23.64	24.09
	p-value: 0.001 (6 runs)	
Setup2	26.98	27.21
	p-value: 0.080 (3 runs)	

Table 6: Results for the Transformer with either multi-head self-attention or proposed MTSA. The reported BLEU values for Setup 1 and 2 are the mean of 5 and 3 runs respectively.

For **Setup1**, we use default hyperparameter set of *transformer_base_single_gpu* provided by official implementation with $1 \times$ P100 , batch size of 2048 and training step of 250K, and report BLEU value for the last checkpoint. For **Setup2**, we use the hyperparameter set of *transformer_base* with the modification of 1) using $4\times$ instead of $8 \times$ P100, 2) increasing batch size from 4096 to 6144 per GPU, and 3) using training step of 133K.

As shown in Table 6, with small p-value for both training setup 1 and 2, the encoder with MTSA significantly outperforms that with multi-head self-attention, which demonstrates that multi-dim based MTSA modeling both pairwise and global dependencies is more expressive than dot-product based multi-head self-attention. Although the results do not improve state-of-the-art BLEU value of machine translation task, the purpose of this experiment to verify the effectiveness of MTSA in contrast to dot-product based multi-head self-attention is accomplished.

5 Conclusion

In conclusion, MTSA is highly parallelizable with more expressive power since it efficiently captures the pairwise dependency at token level, but delicately models the global dependency at feature level, and distributes computations to multiple heads, each equipped with a distinct positional mask. These lead to a sweet spot of the trade-off between performance and efficiency, and make MTSA as memory-efficient as CNN and scalable to long sequences but outperform previous (and even multi-dim) self-attention mechanisms in terms of prediction quality. The experiments conducted on nine NLP tasks verify that the MTSA can reach state-of-the-art performance with appealing efficiency.

Acknowledgments

This research was funded by the Australian Government through the Australian Research Council (ARC) under grants 1) LP160100630 partnership with Australia Government Department of Health and 2) LP150100671 partnership with Australia Research Alliance for Children and Youth (ARACY) and Global Business College Australia (GBCA). We also acknowledge the support of NVIDIA Corporation and MakeMagic Australia with the donation of GPUs.

References

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

Samuel R Bowman, Gabor Angeli, Christopher Potts, and Christopher D Manning. 2015. A large annotated corpus for learning natural language inference. In *EMNLP*.

Samuel R Bowman, Jon Gauthier, Abhinav Rastogi, Raghav Gupta, Christopher D Manning, and Christopher Potts. 2016. A fast unified model for parsing and sentence understanding. In *ACL*.

Denny Britz, Anna Goldie, Thang Luong, and Quoc Le. 2017. Massive exploration of neural machine translation architectures. *arXiv preprint arXiv:1703.03906*.

Qian Chen, Zhen-Hua Ling, and Xiaodan Zhu. 2018. Enhancing sentence embedding with generalized pooling. *arXiv preprint arXiv:1806.09828*.

Qian Chen, Xiaodan Zhu, Zhen-Hua Ling, Si Wei, Hui Jiang, and Diana Inkpen. 2017. Recurrent neural network-based sentence encoder with gated attention for natural language inference. In *RepEval@ EMNLP*.

Jihun Choi, Kang Min Yoo, and Sang goo Lee. 2018. Learning to compose task-specific tree structures. In *AAAI*.

Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. In *NIPS*.

Yann N Dauphin, Angela Fan, Michael Auli, and David Grangier. 2016. Language modeling with gated convolutional networks. *arXiv preprint arXiv:1612.08083*.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N Dauphin. 2017. Convolutional sequence to sequence learning. *arXiv preprint arXiv:1705.03122*.

Alex Graves, Navdeep Jaitly, and Abdel-rahman Mohamed. 2013. Hybrid speech recognition with deep bidirectional lstm. In *Automatic Speech Recognition and Understanding (ASRU), 2013 IEEE Workshop on*, pages 273–278. IEEE.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *CVPR*.

Luheng He, Kenton Lee, Omer Levy, and Luke Zettlemoyer. 2018. Jointly predicting predicates and arguments in neural semantic role labeling. In *ACL*.

Luheng He, Kenton Lee, Mike Lewis, and Luke Zettlemoyer. 2017. Deep semantic role labeling: What works and what?s next. In *ACL*.

Minghao Hu, Yuxing Peng, and Xipeng Qiu. 2017. Reinforced mnemonic reader for machine comprehension. *CoRR, abs/1705.02798*.

Minqing Hu and Bing Liu. 2004. Mining and summarizing customer reviews. In *SIGKDD*.

Jinbae Im and Sungzoon Cho. 2017. Distance-based self-attention network for natural language inference. *arXiv preprint arXiv:1712.02047*.

Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. 2014. A convolutional neural network for modelling sentences. *arXiv preprint arXiv:1404.2188*.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *EMNLP*.

Ryan Kiros, Yukun Zhu, Ruslan R Salakhutdinov, Richard Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Skip-thought vectors. In *NIPS*.

Tao Lei and Yu Zhang. 2017. Training rnns as fast as cnns. *arXiv preprint arXiv:1709.02755*.

Xin Li and Dan Roth. 2002. Learning question classifiers. In *ACL*.

Zheng Li, Ying Wei, Yu Zhang, and Qiang Yang. 2018. Hierarchical attention transfer network for cross-domain sentiment classification. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Zhouhan Lin, Minwei Feng, Cicero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio. 2017. A structured self-attentive sentence embedding. In *ICLR*.

Yang Liu, Chengjie Sun, Lei Lin, and Xiaolong Wang. 2016. Learning natural language inference using bidirectional lstm model and inner-attention. *arXiv preprint arXiv:1605.09090*.

Minh-Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Effective approaches to attention-based neural machine translation. In *EMNLP*.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Yixin Nie and Mohit Bansal. 2017. Shortcut-stacked sentence encoders for multi-domain inference. *arXiv preprint arXiv:1708.02312*.

Bo Pang and Lillian Lee. 2004. A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. In *ACL*.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training. *OpenAI Research*.

Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. 2017. Bidirectional attention flow for machine comprehension. In *ICLR*.

Tao Shen, Tianyi Zhou, Guodong Long, Jing Jiang, Shirui Pan, and Chengqi Zhang. 2018a. Disan: Directional self-attention network for rnn/cnn-free language understanding. In *AAAI*.

Tao Shen, Tianyi Zhou, Guodong Long, Jing Jiang, Sen Wang, and Chengqi Zhang. 2018b. Reinforced self-attention network: a hybrid of hard and soft attention for sequence modeling. *arXiv preprint arXiv:1801.10296*.

Tao Shen, Tianyi Zhou, Guodong Long, Jing Jiang, and Chengqi Zhang. 2018c. Bi-directional block self-attention for fast and memory-efficient sequence modeling. In *ICLR*.

Richard Socher, Alex Perelygin, Jean Y Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, Christopher Potts, et al. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *EMNLP*.

Emma Strubell, Patrick Verga, Daniel Andor, David Weiss, and Andrew McCallum. 2018. Linguistically-informed self-attention for semantic role labeling. *arXiv preprint arXiv:1804.08199*.

Oscar Täckström, Kuzman Ganchev, and Dipanjan Das. 2015. Efficient inference and structured learning for semantic role labeling. *Transactions of the Association for Computational Linguistics*, 3:29–41.

Aarne Talman, Anssi Yli-Jyra, and Jorg Tiedemann. 2018. Natural language inference with hierarchical bilstm max pooling architecture. *arXiv preprint arXiv:1808.08762*.

Zhixing Tan, Mingxuan Wang, Jun Xie, Yidong Chen, and Xiaodong Shi. 2017. Deep semantic role labeling with self-attention. *arXiv preprint arXiv:1712.01586*.

Ashish Vaswani, Shazeer, Noam, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *NIPS*.

Janyce Wiebe, Theresa Wilson, and Claire Cardie. 2005. Annotating expressions of opinions and emotions in language. *Language resources and evaluation*, 39(2):165–210.

Adina Williams, Nikita Nangia, and Samuel R Bowman. 2017. A broad-coverage challenge corpus for sentence understanding through inference. *arXiv preprint arXiv:1704.05426*.

Adams Wei Yu, David Dohan, Minh-Thang Luong, Rui Zhao, Kai Chen, Mohammad Norouzi, and Quoc V Le. 2018. Qanet: Combining local convolution with global self-attention for reading comprehension. In *ICLR*.

Jie Zhou and Wei Xu. 2015. End-to-end learning of semantic role labeling using recurrent neural networks. In *ACL-IJCNLP*.