# Continuous Self-Attention Models with Neural ODE Networks

**Jing Zhang**[1], **Peng Zhang**[1*], **Baiwen Kong**[1], **Junqiu Wei**[2], **Xin Jiang**[2]
[1]College of Intelligence and Computing, Tianjin University, Tianjin, China
[2]Huawei Noah's Ark Lab, China
{zhang_jing, pzhang, bwkong}@tju.edu.cn
{weijunqiu, Jiang.Xin}@huawei.com

## Abstract

Stacked self-attention models receive widespread attention, due to its ability of capturing global dependency among words. However, the stacking of many layers and components generates huge parameters, leading to low parameter efficiency. In response to this issue, we propose a lightweight architecture named *Continuous Self-Attention models with neural ODE networks* (CSAODE). In CSAODE, continuous dynamical models (i.e., neural ODEs) are coupled with our proposed self-attention block to form a self-attention ODE solver. This solver continuously calculates and optimizes the hidden states via only one layer of parameters to improve the parameter efficiency. In addition, we design a novel accelerated continuous dynamical model to reduce computing costs, and integrate it in CSAODE. Moreover, since the original self-attention ignores local information, CSAODE makes use of N-gram convolution to encode local representations, and a fusion layer with only two trainable scalars are designed for generating sentence vectors. We perform a series of experiments on text classification, natural language inference (NLI) and text matching tasks. With fewer parameters, CSAODE outperforms state-of-the-art models on text classification tasks (e.g., 1.3% accuracy improved on SUBJ task), and has competitive performances for NLI and text matching tasks as well.

## Introduction

Transformer has achieved successful performances in many natural language processing (NLP) tasks (Vaswani et al. 2017; Shen et al. 2018b,c; Dai, Li, and Xu 2020), since its self-attention mechanism can effectively model the global dependencies among words (Shen et al. 2017; Vaswani et al. 2017). However, self-attention networks have a limitation in its stacked structures. A *single-layer* self-attention component cannot fully learn the information in the text representations. To capture more abstract features, positional information and so on, the self-attention components are usually stacked, leading to the parameter redundancy (Dehghani et al. 2018).

On the other hand, based on continuous dynamical systems, neural ordinary differential equations (Neural ODEs) (Chen et al. 2018) are a novel family of deep neural network (NN) models, in which the continuous hidden states can be learned using only *one layer* of parameters. In supervised learning tasks (e.g., image classification), Neural ODEs (e.g., RKNet and RKFNet) (Chen et al. 2018), with a small number of parameters, can obtain similar accuracy scores as the six-layer stacked ResNet (He et al. 2015).

This inspires us to solve the parameter deficiency problem of the stacked self-attention models, by proposing a *Continuous Self-Attention models with neural ODE networks* (CSAODE). In our proposed CSAODE, we first design a self-attention block which includes a *Self-Attention Layer* and a *Encoding with Position* component, which can effectively learn global features with position information. Then, we represent such a self-attention block by a continuous self-attention system, in which continuous hidden states are calculated by the self-attention ODE solver.

CSAODE is capable of learning the continuous hidden states of the representation matrix with only one layer of parameters, for performing multiple calculations and optimizations, and finally obtaining the global representation matrix. Moreover, the self-attention ODE solver calculates the gradients of parameters by the adjoint ODE back-propagation approach (Chen et al. 2018), which can be integrated into the end-to-end CSAODE model.

In addition, to reduce the computational costs in existing continuous dynamical models RKNet and RKFNet (Chen et al. 2018), we design a novel continuous dynamical model named as ARKNet, by proposing an accelerated Runge-Kutta method (Udwadia and Farahani 2008), which improves the training speed compared to RKNet. Then we couple such an ARKNet and our proposed self-attention block to construct a self-attention ARK solver in CSAODE.

In addition to the global dependencies modeled in the original self-attention network, the local dependencies are also important for the sequence modeling. Therefore, in CSAODE, we introduce a local representation matrix calculated by N-Gram convolution, which will be inputted to the self-attention ODE solver. Furthermore, we develop a fusion layer consisting of *Euclidean-Attention* and *Gate*. The *Euclidean-Attention* computes the absolute difference between the local representation and the global representation, and such a *Gate* can fuse all features by two trainable scalars.

To sum up, the contributions of our work are:

[*]Corresponding Author: Peng Zhang

- We propose the *Self-Attention ODE Solver* coupling existing continuous dynamical models (RKNet ,RKFNet) and our self-attention block. The solver effectively learns continuous hidden states of features with positional information, and train the global representation matrix via high parameter efficiency.
- We design an accelerated RKNet (ARKNet) coupling with our proposed self-attention block to construct the *Self-Attention ARK Solver*. The ARK solver has similar accuracy scores as the RK solver, with faster training speed.
- CSAODE can learn local dependencies by *N-Gram Convolution* and global-range features by *Self-Attention ODE Solver*, and its fusion layer can effectively balance global and local features by very few parameters (two scalars).
- In general, CSAODE obtains competitive experimental results, with less than one million parameters (much less than the other models). In particular, CSAODE can achieve better test accuracy than a state-of-the-art lightweight model (Dai, Li, and Xu 2020) on five text classification tasks, i.e., TREC (+0.6%), SUBJ (+1.3%), CR (+1.6%), MPQA (+0.4%), and MR (+0.1%).

## Related Work

Self-attention networks usually need to be stacked to improve results, such as the (Vaswani et al. 2017) and (Guo, Zhang, and Liu 2019). However, the stacking of self-attention usually causes the parameter redundancy. This issue has attracted many researchers to study.

### Self-Attention Networks

For the issue of stacking, the universal transformer (UT) (Dehghani et al. 2018) introduces a parallel-in-time self-attentive recurrent sequence model, which designs a parameter sharing mechanism between different stacked layers, and embeds discrete time variables to represent the depth of layers. However, for recurrent mechanism of RNN, the gradient will decay exponentially as the time increases. Due to the application of the RNN mechanism, UT has a performance degrading problem as the depth increases (Bai, Kolter, and Koltun 2019). Moreover, in the ALBERT model (Lan et al. 2019), its parameter sharing method plays the role of parameter compression, while the effect on the results is negative.

For feature extraction tasks such as text classification and matching, there are some lightweight models based on the self-attention structures (Yang et al. 2018, 2019). Shen et al. (2018b) develop the Reinforced Self-Attention Network (ReSAN), in which self-attention is used to model the sparse dependencies between the head and dependent tokens by reinforcement learning. By a mask mechanism, Dai, Li, and Xu (2020) propose MPSAN model which uses different self-attention components to capture temporal order and positional information, respectively. However, for non-sequential modeling tasks, the temporal order and positional information may not be necessarily learned by multiple self-attention components.

### Neural ODEs

Neural ODEs (Chen et al. 2018) consists of RKNet with a stable step size by Runge–Kutta (RK) ODE numerical solution (Jameson, Schmidt, and Turkel 1981) and RKFNet with an adaptive step size by the Runge–Kutta–Fehlberg (RKF) algorithm (Fehlberg 1969). The neural ODE networks with high parameter efficiency can be viewed as a versatile continuous version of a ResNet (He et al. 2015), which can model many kinds of dynamical systems (Liu et al. 2020). In addition to high parameter efficiency of deep neural networks, Neural ODEs also have other advantages such as memory efficiency and continuous time-series models.

As mentioned before, Neural ODEs have attracted many researchers to study. Voelker, Kajić, and Eliasmith (2019) develop Legendre ODEs continuous memory units to couple LSTM networks, learning the dependent relationship spanning 100,000 time steps without exponential decay of gradients. GRU-ODE-Bayes (De Brouwer et al. 2019) improves GRU models by ODEs mechanism, adding Bayes algorithms to achieve the continuous time modeling of sporadic and irregular data in the real world. The human attention prior modeled by continuous dynamics helps the machine to implicitly ignore unnecessary reasoning steps and find the shortest reasoning path in (Kim and Lee 2019).

It is worth noting that there are two main types of continuous dynamical models: RKNet and RKFNet. The RKFNet is good at series modeling of uneven time with adaptive step size, but the training speed drops sharply as the step size increases. For RKFNet, a recent work (Dupont, Doucet, and Teh 2019) solves the problem of the explosion of functional evaluations in RKFNet. On the contrary, for RKNet, the problem of training speed remains unsolved.

## The Architecture of CSAODE

As illustrated in Figure 1, in the proposed Continuous Self-Attention models with neural ODE networks (CSAODE), we first come up with a self-attention ODE solver which has high parameter efficiency. The solver continuously calculates and optimizes the output of our designed self-attention block via parameters of one block. Moreover, in order to improve training speed, the self-attention accelerated Runge-Kutta solver is also proposed in our work. Finally, we use a simple N-gram convolution to capture local dependency of texts, and we design a fusion layer to fuse all features with two parameters. As mentioned above, the CSAODE is a lightweight architecture.

### N-Gram Convolution

The input of CSAODE is a sequence $S$ with $M$ words $[w_1, \dots, w_i, \dots, w_M]$. We use a look-up layer to transform a word $w_i$ into a word embedding $\boldsymbol{h}_i \in \mathbb{R}^{d_e}$, where $d_e$ is the dimension of word embeddings. The representation matrix $\boldsymbol{H} \in \mathbb{R}^{M \times d_e}$ of the sequence $S$ can be written as:

$$\boldsymbol{H} = [\boldsymbol{h}_1, \boldsymbol{h}_2, \dots, \boldsymbol{h}_M] \tag{1}$$

Then, we generate the local representation $\hat{\boldsymbol{h}}_j \in \mathbb{R}^d$ for the phrase $\boldsymbol{R}_j = [\boldsymbol{h}_j, \dots, \boldsymbol{h}_{j+N-1}] \in \mathbb{R}^{N \times d_e}$, $0 \leq j \leq$

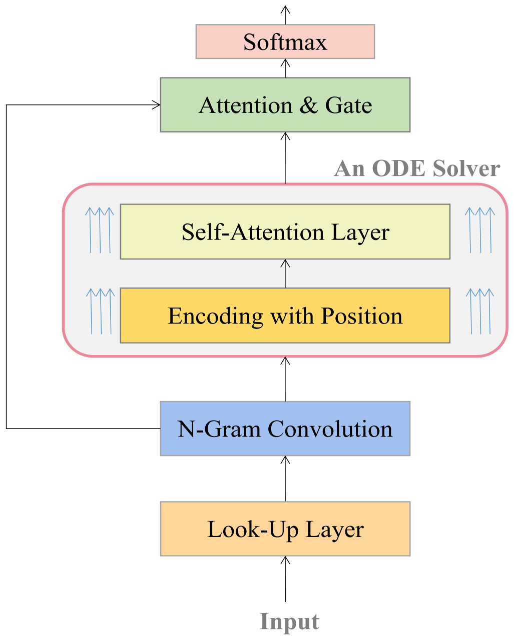


Figure 1: The Architecture of CSAODE

$M-N+1$, using the convolution weights $\boldsymbol{W}_L \in \mathbb{R}^{Nd_e \times d}$:

$$\hat{\boldsymbol{h}}_j = \text{ReLU}(\boldsymbol{R}_j \boldsymbol{W}_L + \boldsymbol{b}_L) \tag{2}$$

where $\boldsymbol{b}_L \in \mathbb{R}^d$, $d$ is the number of feature maps, and ReLU is activation function. Then, $\hat{\boldsymbol{H}} = [\hat{\boldsymbol{h}}_1, \dots, \hat{\boldsymbol{h}}_{M-N+1}] \in \mathbb{R}^{(M-N+1)\times d}$ is padded by zero vectors to form a local representation matrix $\boldsymbol{H}^L \in \mathbb{R}^{M\times d}$.

## The Self-Attention ODE Solver

In the self-attention ODE solver shown in Figure 2, we propose a basic neural network component (i.e., a self-attention block), which consists of *Encoding with Position* and *Self-Attention Layer* structures.

**Encoding with Position** In *Encoding with Position*, the representation $\boldsymbol{P} \in \mathbb{R}^{M\times d}$ of position encoding is fixed in the training process, and it is defined as:

$$\boldsymbol{P}_{i,j} = \begin{cases} \sin(i \cdot c^{\frac{j}{d}}) & j \in even, \\ \cos(i \cdot c^{\frac{j-1}{d}}) & j \in odd, \end{cases} \tag{3}$$

where $i \in \{1, \dots, M\}$ is the positional index of phrase features in a sequence, and $j \in \{1, \dots, d\}$ is the dimensional index of a phrase vector. In addition, the best value of $c$ is $10^{-4}$ in (Vaswani et al. 2017).

Then, the positional information $\boldsymbol{P}$ is added to the representation matrix $\boldsymbol{H}^L$. Finally, $\boldsymbol{H}^L + \boldsymbol{P}$ will be learned by a linear transformation, which apply on ReLU and Norm functions as shown below:

$$\boldsymbol{x} = \text{ReLU}(\text{Norm}((\boldsymbol{H}^L + \boldsymbol{P})\boldsymbol{W}_P + \boldsymbol{b}_P)) \tag{4}$$

where $\boldsymbol{W}_P \in \mathbb{R}^{d\times d}, \boldsymbol{b}_P \in \mathbb{R}^d$, and Norm represents a layer normalization function.

**Self-Attention Layer** With the same input $\boldsymbol{x}$, the $\boldsymbol{Q}$,$\boldsymbol{K}$ and $\boldsymbol{V}$ in self-attention models can be computed. Then, by the dot-product attention and activation, the output $\boldsymbol{G}$ is:

$$\begin{aligned} \boldsymbol{G} &= \text{ReLU}(\text{Norm}(\boldsymbol{A} \cdot \boldsymbol{x}\boldsymbol{W}_V)) \\ \boldsymbol{A} &= \text{softmax}(\boldsymbol{x}\boldsymbol{W}_Q \cdot (\boldsymbol{x}\boldsymbol{W}_K)^\top) \end{aligned} \tag{5}$$

where $\boldsymbol{x}\boldsymbol{W}_Q$, $\boldsymbol{x}\boldsymbol{W}_K$ and $\boldsymbol{x}\boldsymbol{W}_V$ represent the $\boldsymbol{Q}$, $\boldsymbol{K}$ and $\boldsymbol{V}$ in the self-attention mechanism, respectively, and $\boldsymbol{W}_Q, \boldsymbol{W}_K, \boldsymbol{W}_K \in \mathbb{R}^{d\times d}$. The self-attention score matrix $\boldsymbol{A}$ is normalized by softmax function. In addition, $\cdot$ represents the dot product and $\top$ is the the transpose operation.

The self-attention block formed by *Encoding with Position* and *Self-Attention Layer* components can be represented as $F(\boldsymbol{H}^L; \boldsymbol{\theta})$, where $\boldsymbol{\theta}$ represents all the trained parameters (e.g., $\boldsymbol{W}_Q$ or $\boldsymbol{b}_P$), and $\boldsymbol{H}^L$ is the input.

**The Self-Attention Runge-Kutta Solver** To improve the parameter efficiency of a self-attention block, continuous dynamical models are used to couple a self-attention block.

First, we design a continuous self-attention system, the formula of which can be written as :

$$\begin{aligned} \boldsymbol{G}(b) = &\boldsymbol{G}(s) + \int_s^b F(\boldsymbol{G}(t), t; \boldsymbol{\theta}) dt \\ &0 \le s \le b \le T \end{aligned} \tag{6}$$

which is with an initial input $\boldsymbol{G}(0) = \boldsymbol{H}^L$, where $\boldsymbol{G}(b)$ and $\boldsymbol{G}(s) \in \mathbb{R}^{M\times d}$ are hidden states at the $b$ and $s$ time steps, and $F(\boldsymbol{G}(t), t; \boldsymbol{\theta})$ is a self-attention block parameterized by $\boldsymbol{\theta}$ and takes the previous state $(\boldsymbol{G}(t), t)$. The final hidden state is $\boldsymbol{G}(T)$ at time $T$, and the global representation matrix $\boldsymbol{H}^G = \boldsymbol{G}(T)$.

The Equation (6) is equivalent to the initial value problem of ODE $\frac{d\boldsymbol{G}(t)}{dt} = F(\boldsymbol{G}(t), t; \boldsymbol{\theta})$, which is guaranteed to have a unique solution under mild conditions (Tenenbaum and Pollard 1963). In addition, the reason why we use the encoding method shown as the Equation (3) is that it can be regarded as a constant position signal to influence each hidden state in ODE, $\frac{d(\boldsymbol{G}(t)+\boldsymbol{P})}{dt} = F(\boldsymbol{G}(t) + \boldsymbol{P}, t; \boldsymbol{\theta})$. The ODE is rewritten as:

$$\frac{d\boldsymbol{G}(t)}{dt} = F(\boldsymbol{G}(t) + \boldsymbol{P}, t; \boldsymbol{\theta}) \tag{7}$$

Then, the continuous dynamical models RKFNet and RKNet (Chen et al. 2018) are used to design the self-attention RKF solver and the self-attention RK solver respectively, to actually calculate the continuous hidden states in Equation (6). In CSAODE, taking the RKNet as an example, the self-attention RK solver can be written as:

$$\begin{aligned} \boldsymbol{G}(t_i) &= \boldsymbol{G}(t_{i-1}) + r\sum_{j=1}^{4} c_j \boldsymbol{Z}_j \quad 1 \le i \le T \\ \boldsymbol{Z}_j &= F(t_{i-1} + a_j r, \boldsymbol{G}(t_{i-1}) + \boldsymbol{P} + a_j r \boldsymbol{Z}_{j-1}; \boldsymbol{\theta}) \end{aligned} \tag{8}$$

where $0 < r \le 1.0$ is a stable step size (i.e., a hyperparameter), $t_i - t_{i-1} = r$, $c_j$ and $a_j$ are also hyperparameters. The input of the solver is $\boldsymbol{G}(t_0) = \boldsymbol{H}^L(t_0 = 0)$, the output is $\boldsymbol{H}^G = \boldsymbol{G}(t_T)$ $(T = \lceil \frac{1}{r} \rceil)$, and $\lceil\rceil$ indicates

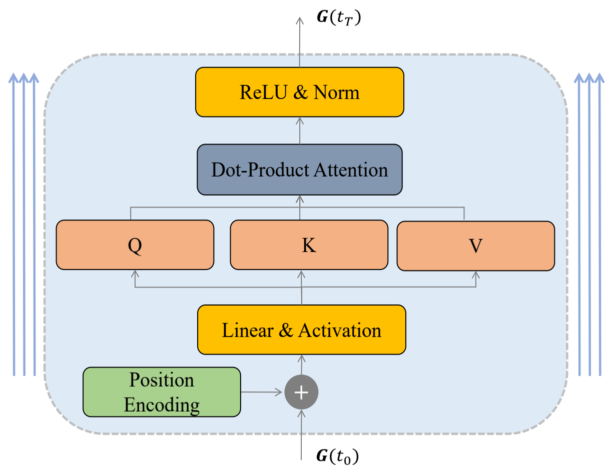


Figure 2: The Self-Attention ODE Solver.

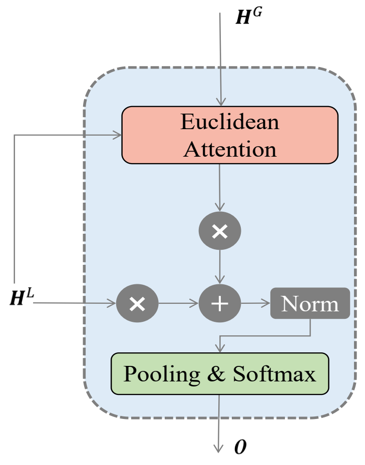


Figure 3: The Fusion Layer.

rounding In addition, $\boldsymbol{Z}_j$ represents the output of the self-attention block $F(\cdot)$ in different states of a time step.

We can make use of the efficient approach proposed by (Chen et al. 2018) to calculate the gradients of $\theta$ with respect to the overall training loss, which allows us to include the parameterized dynamical self-attention block into the end-to-end training of CSAODE.

## The Self-Attention Accelerated-RK Solver

After successfully coupling existing continuous dynamical models (e.g., RKNet) with our self-attention block to form the self-attention ODE solver written as Equation (8), we propose a new continuous dynamical model named ARKNet via accelerated Runge-Kutta (ARK) algorithm (Udwadia and Farahani 2008), which has a truncation error of $O(r^5)$ as RKNet. This can be demonstrated in Claim 1.

**Claim 1** *The order of local truncation error of the self-attention accelerated Runge-Kutta (ARK) solver shown as Equation (9) is $O(r^5)$, which is the same as the self-attention Runge-Kutta (RK) solver written as Equation (8).*

The same truncation error means ARKNet and RKNet have similar accuracy, while the training speed of ARKNet is higher than RKNet. The self-attention ARK solver formed by a self-attention block and ARKNet is written as:

$$\begin{aligned}\boldsymbol{G}(t_{i+1}) =& c_0\boldsymbol{G}(t_i) - c_{-0}\boldsymbol{G}(t_{i-1}) + c_1\boldsymbol{Z}_1 - c_{-1}\boldsymbol{Z}_{-1} \\ &+ \sum_{j=2}^{3} c_j(\boldsymbol{Z}_j - \boldsymbol{Z}_{-j}) \quad 1 \le i \le T-1\end{aligned} \tag{9}$$

where $c_0, c_{-0}$ and other coefficients are hyper-parameters. $\boldsymbol{G}(t_0) = \boldsymbol{H}^L$ $(t_0 = 0)$ is the input, the output is $\boldsymbol{H}^G = \boldsymbol{G}(t_T)$ $(T = \lceil \frac{1}{r} \rceil)$, and $\boldsymbol{G}(t_{i+1}) \in \mathbb{R}^{M\times d}$ is the hidden state at the $t_{i+1}$ time step.

It is worth noting that the self-attention ARK solver shown as Equation (9) cannot be self-started, that is, $\boldsymbol{G}(t_1)$ is computed by Equation (8). When the step size $r = 1.0$, the Equation (9) will not be calculated, so that $0 < r < 1$. Moreover, $\boldsymbol{Z}_j$ or $\boldsymbol{Z}_{-j}$ represent the outputs of the self-attention block $F(\cdot)$ in different states of a time step:

$$\begin{aligned}\boldsymbol{Z}_j &= rF(t_i + a_j r, \boldsymbol{G}(t_i) + \boldsymbol{P} + a_j r\boldsymbol{Z}_{j-1}; \boldsymbol{\theta}) \\ \boldsymbol{Z}_{-j} &= rF(t_i + a_j r, \boldsymbol{G}(t_{i-1}) + \boldsymbol{P} + a_j r\boldsymbol{Z}_{-j+1}; \boldsymbol{\theta})\end{aligned} \tag{10}$$

where $a_j$ $(j \in \{2, 3\})$ are the stable hyper-parameters which can be calculated by accelerated Runge-Kutta algorithm.

## Attention and Gate Layer

After the global representation matrix $\boldsymbol{H}^G$ and local representation matrix $\boldsymbol{H}^L$ of a sequence are obtained, the fusion layer is illustrated in Figure 3. The fusion layer consists of *Euclidean-Attention* and *Gate*, and compares $\boldsymbol{H}^L$ and $\boldsymbol{H}^G$ to perform the feature fusion.

**Euclidean-Attention** For obtaining the absolute difference (Yin et al. 2016) between global features and local features, we use Euclidean distance to compute a Euclidean-attention score matrix $\boldsymbol{E} \in \mathbb{R}^{M\times M}$, which is written as:

$$\boldsymbol{E}_{lg} = \frac{1}{\sqrt{\sum_{j=1}^{d}(\boldsymbol{h}_{lj}^L - \boldsymbol{h}_{gj}^G)^2}} \tag{11}$$

$$\boldsymbol{H}^E = \boldsymbol{E} \cdot \boldsymbol{H}^L \tag{12}$$

where $\boldsymbol{H}^L = [\boldsymbol{h}_1^L, \ldots, \boldsymbol{h}_l^L, \ldots, \boldsymbol{h}_M^L]$, the global representation matrix $\boldsymbol{H}^G = [\boldsymbol{h}_1^G, \ldots, \boldsymbol{h}_g^G, \ldots, \boldsymbol{h}_M^G]$, and $d$ is the dimension of these vectors. The attention scores between different $\boldsymbol{h}_l^L$ and $\boldsymbol{h}_g^G$ vectors are reciprocals of Euclidean distance, which means that the closer the distance, the higher the score. By using the dot-product, the attention score matrix $\boldsymbol{E}$ is applied in $\boldsymbol{H}^L$ to make global features focus on important local features, which is $\boldsymbol{H}^E \in \mathbb{R}^{M\times d}$.

**Gate** We use trainable parameters to construct *Gate*, which chooses more effective information in $\boldsymbol{H}^E$ and $\boldsymbol{H}^L$ to calculate the fused representation matrix $\boldsymbol{H}^O \in \mathbb{R}^{M\times d}$:

$$\boldsymbol{H}^O = \mathrm{Norm}(W_1 \times \boldsymbol{H}^L + W_2 \times \boldsymbol{H}^E) \tag{13}$$

which is optimized by layer normalization function Norm, where $W_1$ and $W_2$ are the parameterized scalars trained in

| Model | CR | MR | MPQA | SUBJ | TREC | $\lvert\theta\rvert$ |
|---|---|---|---|---|---|---|
| CNN (Kim 2014) | 84.7 | 81.0 | 89.6 | 93.0 | 92.8 | 1.40M |
| Multi-head (Vaswani et al. 2017) | 82.6 | – | 89.8 | 94 | 93.4 | 2.00M |
| SA-SNN (Zhao et al. 2018) | – | 82.1 | – | 93.9 | – | 3.75M |
| DiSAN (Shen et al. 2017) | 84.8 | – | 90.1 | 94.2 | 94.2 | 2.35M |
| Bi-BloSAN (Shen et al. 2018c) | 84.8 | – | 90.4 | 94.5 | 94.8 | 2.80M |
| MPSAN (Dai, Li, and Xu 2020) | 85.4 | – | 90.4 | 94.6 | 94.8 | 1.09M |
| CSARKF | 86.5 | 82.1 | **90.8** | 95.2 | **95.4** | 0.63M |
| CSARK | **87** | 81.6 | 90.5 | **95.9** | 94.8 | 0.63M |
| CSAARK | 86.2 | **82.2** | 90.5 | 95.2 | 95.2 | 0.63M |

Table 1: Experimental results (Accuracy) on the test sets of CR, MR, MPQA, SUBJ and TREC classification tasks. The $\theta$ represents the number of network parameters. "–" means results are not reported.

the CSAODE. Finally, the output $\boldsymbol{o}$ of classifier is written as:

$$\boldsymbol{o} = \text{pooling}(\boldsymbol{H}^O)\boldsymbol{W}_O + \boldsymbol{b}_O \tag{14}$$

where $\boldsymbol{o} \in \mathbb{R}^{d_l}$, pooling is the max-pooling function, $\boldsymbol{W}_O \in \mathbb{R}^{d \times d_l}$ and $\boldsymbol{b}_O \in \mathbb{R}^{d_l}$ are the parameters of a FFN layer, and the $d_l$ is the number of classes.

## Experiments

We test our CSAODE models on text classification tasks, natural language inference and question-answering tasks. Moreover, our CSAODE architectures are implemented with RKFNet, RKNet and ARKNet, which are then denoted by CSARKF, CSARK and CSAARK, respectively.

For all tasks, we implement our model with Pytorch-1.20, and train them on a Nvidia P40 GPU. Word embeddings are initialized by GloVe (Pennington, Socher, and Manning 2014) with 300-dimension. All other parameters are initialized with Xavier (Glorot and Bengio 2010) and normalized by weight normalization (Salimans and Kingma 2016). As for learning method, we use the Adam optimizer (Kingma and Ba 2014) and an exponentially decaying learning rate with a linear warm up. The dimension of the hidden vectors is set to 300, which is equal to the word embedding size. As for convolution, the filter size is set to 2. In addition, dropout with a keep probability of 0.1 is applied in the layers. The initial learning rate is set from 0.0001 to 0.003 and the batch size is tuned from 80 to 256. The L2 regularization decay factor is $10^{-5}$. In addition, the initial step size of the self-attention ODE solver is tuned from $10^{-2}$ to $5 \times 10^{-1}$.

### Text Classification

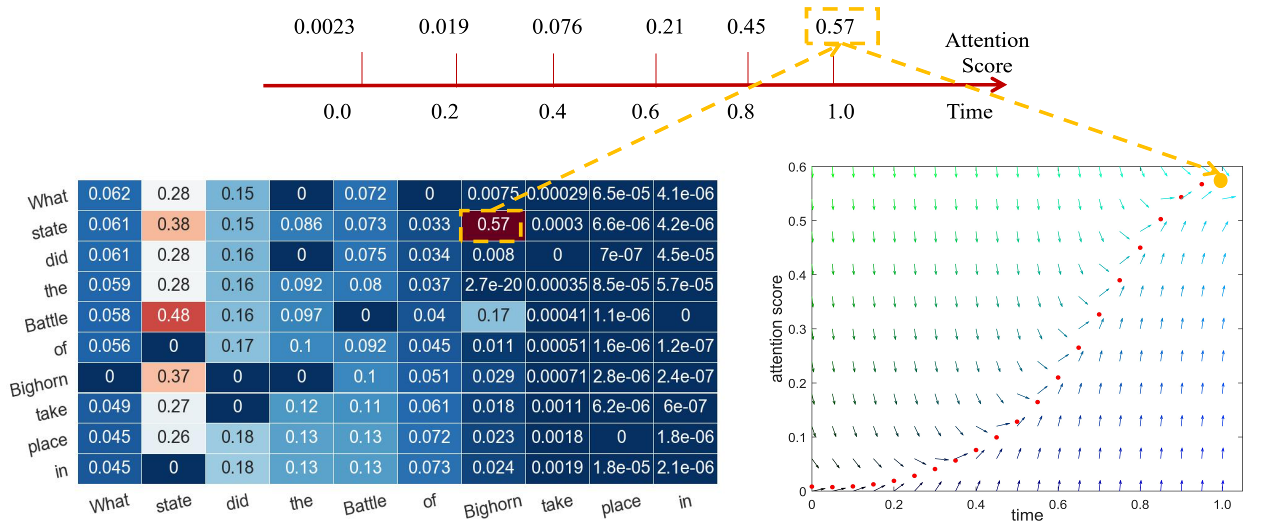
We evaluate our model on five text classification tasks. MR (Pang and Lee 2004a): Movie reviews are divided into positive and negative categories; CR (Hu and Liu 2004): Customer reviews set where the task is to predict positive or negative product reviews; SUBJ (Pang and Lee 2004b): Subjectivity dataset where the target is to classify a text as being subjective or objective; MPQA (Wiebe, Wilson, and Cardie 2005): Opinion polarity detection subtask; TREC (Li and Roth 2002): question classification dataset which involves classifying a question into 6 question types. We choose small-scale baselines based on CNN or self-attention components. Accuracy is used as the evaluation metric.

As shown in Table 1, compared with the SOTA results (achieved by MPSAN), CSARKF improves the test accuracy by 0.4%, 0.6% for MPQA and TREC tasks, respectively. Moreover, CR and SUBJ tasks are improved by 1.6% and 1.3% respectively by CSARK. In addition, the test accuracy of MR increases 0.1% via CSAARK. It is worth noting that CSAODE models (i.e., CSARKF, CSARK and CSAARK) have only about 630,000 parameters, and compared with CNN, Multi-head, SA-SNN, DiSAN, Bi-BloSAN and MPSAN, CSAODE models reduce the parameters by 0.77M, 1.37M, 3.12M, 1.72M, 2.17M and 0.46M, respectively. We found that for most tasks, the performance of CSAARK is weaker than CSARK, because CSAARK loses a small amount of accuracy while reducing the training cost.

**Continuity Analysis** The continuity analysis is based on the self-attention scores changing over time step.

In the experiments, the window size of convolution is set to 1, so the tokens of self-attention score matrices in the self-attention ODE solvers can represent words in a sentence. In addition, the step size $r$ of solvers is equal to 0.2, and we record the self-attention score matrices of some time steps from 0.0 to 1.0. As shown in Figure 4, we use the sentence "What state did the Battle of Bighorn take place in" obtained from TREC test set to demonstrate the high continuity in our model. In this sentence, the words "state" and "Bighorn" are closely related, of which the attention should be crucial for determining the answer. In the training, attention score of this pair of words keeps increasing from 0.0023 to 0.57 through a dynamic process, suggesting that important information has been obtained by our model. A vector field graph made from attention scores is also displayed in Figure 4. The smooth curve and surrounding arrows further indicate the continuous nature of the learning process.

**Ablation Experiments** We present an ablation study of our CSAARK model. For the original CSAARK, its step size is set to $2 \times 10^{-1}$ on TREC. This study compares the original CSAARK with three ablation baselines: (1) "w/o ODE (1 block)": without coupling continuous dynamical models, and only use a self-attention block; (2) "w/o ODE (5 blocks)": stacking five self-attention blocks are connected by residual connection, and without ARKNet; (3) "w/o conv": only without *N-Gram Convolution*. We carry out a set of experiments on the TREC task in Table 2.



The state of self-attention score matrix at time 1.0. Vector field plot of the self-attention score over time.

Figure 4: Continuity Analysis. The sub-figure on the left is the heat map of the self-attention score matrix $\boldsymbol{A}$ shown as Equation (5) at time 1.0. The sub-graph at the top shows the states of a certain value in the self-attention score matrix over time. With the records in the top sub-graph, the vector field plot of the self-attention score over time are shown on the right sub-figure.

| Model | Test Acc | $\lvert\theta\rvert$ |
|---|---|---|
| w/o ODE (1 block) | 93.7 | 0.63M |
| w/o ODE (5 blocks) | 94.2 | 1.62M |
| w/o conv | 92.8 | 0.45M |
| CSAARK | 95.2 | 0.63M |

Table 2: An ablation study of CSAARK model on the TREC dataset. $|\theta|$ is the number of parameters.

The first ablation baseline shows that without continuous dynamical models (i.e., ARKNet), the performance on TREC task degrades significantly. The step size of CSAARK is 0.2, which means that the self-attention block is called for five times in a solver. In the second baseline, five self-attention blocks are stacked for comparison with original CSAARK, and the test accuracy increases by 0.5% than the 1 block baseline. However, the result of second baseline still has a 1% gap with CSAARK, while the second baseline adds around 1000,000 parameters. The results of the third baseline show that without n-gram convolution learning local information, the performance of CSAARK has a decrease by 2.4% on TREC task.

**Performance analysis** In this section, we first analyze the time performance between CSARK and CSAARK models on TREC task. Reported results are the average of 10 runs.

As shown in Figure 5, the step sizes of the self-attention solvers are set to 0.1, 0.2 ,0.4, 0.6, 0.8 and 1.0. In the trends of CSAARK and CSARK, with the step size getting smaller, the training time increases. This phenomenon is because as the step size decreases, the time step increases, then the calculation cost increases. In addition, if the step size $r < 1.0$, the time per epoch of CSAARK is continuously lower than

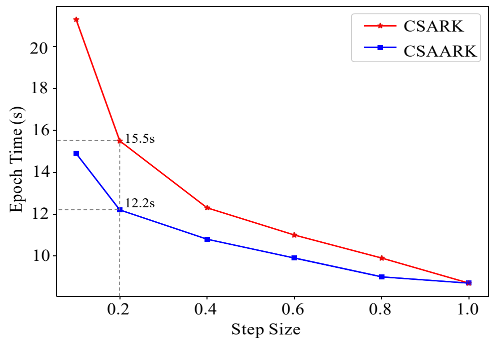


Figure 5: As the step size changes, the training time comparison between CSARK and CSAARK.

CSARK model, which illustrates the effectiveness of accelerated RKNet. Moreover, when the step size $r$ is 1.0, the computation of CSAARK is equivalent to CSARK. As for training speed, when step size $r$ is set to 2e-1, the training speed of CSAARK is 12.2 second per epoch, which is a 21.3% increase compared to CSARK model.

Regarding the influence of the dimension, the convolution window size and step size $r$ in CSAARK model, we perform comparative experiments for the number of parameter ($|\theta|$) and performance (test accuracy). In Table 3, “d” means the dimension $d$ of hidden layers, “cov” means the convolution window size ($N$ in N-gram) and “r” means step size of ODE.

First, CSAARK (300d+2conv+0.2r) gets the best test accuracy of 95.2% in the fourth part of the Table 3. In the first part, the convolution window size of the CSAARK is fixed, the hidden dimension is set to 50 and 200, respectively. It is obvious that as the hidden dimension increases,

| Model | Test Acc | $\lvert\theta\rvert$ |
|---|---|---|
| CSAARK (50d+2cov+0.2r) | 91.6 | 0.043M |
| CSAARK (200d+2conv+0.2r) | 94.3 | 0.32M |
| CSAARK (300d+1cov+0.2r) | 94.1 | 0.54M |
| CSAARK (300d+3cov+0.2r) | 94.5 | 0.72M |
| CSAARK (300d+2conv+0.5r) | 94.6 | 0.63M |
| CSAARK (300d+2conv+0.1r) | 95.0 | 0.63M |
| CSAARK (300d+2conv+0.2r) | 95.2 | 0.63M |

Table 3: Performance analysis experiments about dimension, step size and filter size. $|\theta|$ is the number of parameters.

the parameters and accuracy increase significantly. On the other hand, in the second part of Table 3, we notice that the result of CSAARK (300d+2cov+0.2r) is higher than CSAARK (300d+1cov+0.2r) by 1.1%. However, it does not mean that performance is in positive relationship with the window size, since when we set 3conv, the result is even lower than CSAARK (300d+2conv+0.2r) by 0.7%. In addition, the results of the third and fourth part in Table 3 evaluate that the smaller the step size $r$, the better the performance may not be, and the amount of parameters does not change.

**Visualization Experiments** To better understand what role the fusion layer plays, we perform analyses of visualization experiments for *Euclidean-Attention* and *Gate*.

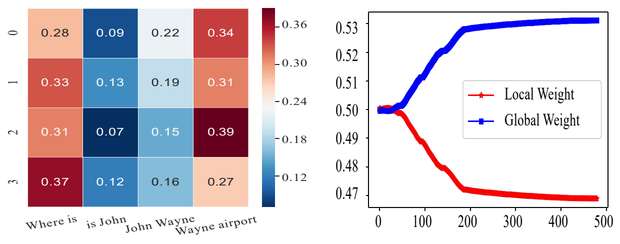
As we can see in the Figure 6(a), the *Euclidean-Attention* is indispensable to help global information capturing important phrase features. For example, a question sentence is the "Where is John Wayne airport" in TREC test set. For "2" tokens, the attention scores for "Where is" and "Wayne airport" are 0.31 and 0.39, respectively, while for "is John" and "John Wayne", the probabilities are only 0.07 and 0.15.

In the Figure 6(b), there are $48 \times 10 = 480$ batches in the training process, where 48 is the number of batches in an epoch, and 10 is the number of epochs. Within 200 epochs, the global weight shows a clear upward trend, while the local weight shows a downward trend from 0.5 to about 0.47. After 200 epochs, the two parameters have a steady trend.

## Natural Language Inference and Text Matching

We first introduce the data sets. SNLI (Bowman et al. 2015): Stanford Natural Language Inference is a benchmark dataset for natural language inference. There are 570k human annotated sentence pairs with four labels, and test accuracy is the metric. WikiQA (Yang, Yih, and Meek 2015) is a retrieval-based question answering dataset based on Wikipedia, which is composed of 20.4k/2.7k/6.2k (train/dev/test) samples. The mean average precision (MAP) and mean reciprocal rank (MRR) are used as the evaluation metrics.

As shown in Table 4, compared with the reported results in recent years, our CSAODE which consists of CSAARK, CSARK and CSARFK models achieves competitive performances for NLI and matching task. On the one hand, the test accuracy of SNLI with our CSAODE is higher than Star-Transformer by 0.1%. On the other hand, we obtain a MRR result similar to CNN+Transformer on the WikiQA dataset.



(a) Euclidean-Attention Scores (b) The Weights of Gate

Figure 6: Visualization Experiments of Fusion Layer. (a) Attention probabilities of Euclidean-Attention in CSAODE. The tokens aligned in horizontal axis are phrase features, and the tokens aligned in vertical axis are global information; (b) Local weight and global weight represent two trainable scalars in Gate, respectively. In the training process of TREC task, the two weights are initialized by 0.5, and always satisfy L1 normalization as the batch changes.

| Model | SNLI | WikiQA | |
|---|---|---|---|
| | Acc | MAP | MRR |
| SWEM (Shen et al. 2018a) | 83.8 | 0.681 | 0.692 |
| Bi-BloSAN (Shen et al. 2018c) | 85.7 | – | – |
| Star-Transformer (Guo et al. 2019) | 86.0 | – | – |
| CNM (Li, Wang, and Melucci 2019) | – | 0.675 | 0.686 |
| LANN (Shao et al. 2019) | – | 0.689 | 0.702 |
| CNN+Transformer (Chen et al. 2020) | – | **0.691** | 0.703 |
| CSAODE | **86.1** | 0.686 | **0.706** |

Table 4: Experimental results on SNLI and WikiQA tasks with CSAODE models.

## Conclusion

In this paper, without stacking, we propose the self-attention ODE solver formed by existing continuous dynamical model (Chen et al. 2018), to effectively calculate the continuous states of hidden features only via one-layer parameters. Moreover, we design a novel accelerated continuous dynamical model coupled in self-attention solver to improve training speed. In addition, the N-gram convolution and fusion layer are used to capture local dependencies and generate sentence representation, respectively. Finally, all mentioned above are integrated in our lightweight CSAODE architecture. The experiment results show that by no more than 1M parameters, our CSAODE achieves good performances on text classification, NLI and matching tasks. In the future, we will apply the continuous dynamical models to a larger architecture such as Transformer (Vaswani et al. 2017), to improve its parameter efficiency.

## Acknowledgments

This work is supported in part by the state key development program of China (grant No.2018YFC0831704), Natural Science Foundation of China (grant No.61772363).

## References

Bai, S.; Kolter, J. Z.; and Koltun, V. 2019. Deep equilibrium models. In *Advances in Neural Information Processing Systems*, 690–701.

Bowman, S. R.; Angeli, G.; Potts, C.; and Manning, C. D. 2015. A large annotated corpus for learning natural language inference. *arXiv preprint arXiv:1508.05326* .

Chen, D.; Peng, S.; Li, K.; Xu, Y.; Zhang, J.; and Xie, X. 2020. Re-Ranking Answer Selection with Similarity Aggregation. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '20, 1677–1680. New York, NY, USA: Association for Computing Machinery. ISBN 9781450380164. doi:10.1145/3397271.3401199. URL https://doi.org/10.1145/3397271.3401199.

Chen, R. T.; Rubanova, Y.; Bettencourt, J.; and Duvenaud, D. K. 2018. Neural ordinary differential equations. In *Advances in neural information processing systems*, 6571–6583.

Dai, B.; Li, J.; and Xu, R. 2020. Multiple Positional Self-Attention Network for Text Classification. In *AAAI*, 7610–7617.

De Brouwer, E.; Simm, J.; Arany, A.; and Moreau, Y. 2019. GRU-ODE-Bayes: Continuous modeling of sporadically-observed time series. In *Advances in Neural Information Processing Systems*, 7379–7390.

Dehghani, M.; Gouws, S.; Vinyals, O.; Uszkoreit, J.; and Kaiser, L. 2018. Universal Transformers. *CoRR* abs/1807.03819. URL http://arxiv.org/abs/1807.03819.

Dupont, E.; Doucet, A.; and Teh, Y. W. 2019. Augmented neural odes. In *Advances in Neural Information Processing Systems*, 3140–3150.

Fehlberg, E. 1969. *Low-order classical Runge-Kutta formulas with stepsize control and their application to some heat transfer problems*. National aeronautics and space administration.

Glorot, X.; and Bengio, Y. 2010. Understanding the difficulty of training deep feedforward neural networks. volume 9 of *Proceedings of Machine Learning Research*, 249–256. Chia Laguna Resort, Sardinia, Italy: JMLR Workshop and Conference Proceedings. URL http://proceedings.mlr.press/v9/glorot10a.html.

Guo, M.; Zhang, Y.; and Liu, T. 2019. Gaussian transformer: a lightweight approach for natural language inference. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, 6489–6496.

Guo, Q.; Qiu, X.; Liu, P.; Shao, Y.; Xue, X.; and Zhang, Z. 2019. Star-Transformer. *CoRR* abs/1902.09113. URL http://arxiv.org/abs/1902.09113.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2015. Deep Residual Learning for Image Recognition. *CoRR* abs/1512.03385. URL http://arxiv.org/abs/1512.03385.

Hu, M.; and Liu, B. 2004. Mining and summarizing customer reviews. In Kim, W.; Kohavi, R.; Gehrke, J.; and DuMouchel, W., eds., *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Seattle, Washington, USA, August 22-25, 2004*, 168–177. ACM. doi:10.1145/1014052.1014073. URL https://doi.org/10.1145/1014052.1014073.

Jameson, A.; Schmidt, W.; and Turkel, E. 1981. Numerical solution of the Euler equations by finite volume methods using Runge Kutta time stepping schemes. In *14th fluid and plasma dynamics conference*, 1259.

Kim, W.; and Lee, Y. 2019. Learning Dynamics of Attention: Human Prior for Interpretable Machine Reasoning. In *Advances in Neural Information Processing Systems*, 6021–6032.

Kim, Y. 2014. Convolutional Neural Networks for Sentence Classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 1746–1751. Doha, Qatar: Association for Computational Linguistics. doi:10.3115/v1/D14-1181. URL https://www.aclweb.org/anthology/D14-1181.

Kingma, D. P.; and Ba, J. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* .

Lan, Z.; Chen, M.; Goodman, S.; Gimpel, K.; Sharma, P.; and Soricut, R. 2019. Albert: A lite bert for self-supervised learning of language representations. *arXiv preprint arXiv:1909.11942* .

Li, Q.; Wang, B.; and Melucci, M. 2019. CNM: An Interpretable Complex-valued Network for Matching. 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)*, 4139–4148. Minneapolis, Minnesota: Association for Computational Linguistics. doi:10.18653/v1/N19-1420. URL https://www.aclweb.org/anthology/N19-1420.

Li, X.; and Roth, D. 2002. Learning question classifiers. In *COLING 2002: The 19th International Conference on Computational Linguistics*.

Liu, X.; Yu, H.-F.; Dhillon, I.; and Hsieh, C.-J. 2020. Learning to Encode Position for Transformer with Continuous Dynamical Model. *arXiv preprint arXiv:2003.09229* .

Pang, B.; and Lee, L. 2004a. A Sentimental Education: Sentiment Analysis Using Subjectivity Summarization Based on Minimum Cuts. In Scott, D.; Daelemans, W.; and Walker, M. A., eds., *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics, 21-26 July, 2004, Barcelona, Spain*, 271–278. ACL. doi:10.3115/1218955.1218990. URL https://www.aclweb.org/anthology/P04-1035/.

Pang, B.; and Lee, L. 2004b. A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. *arXiv preprint cs/0409058* .

Pennington, J.; Socher, R.; and Manning, C. D. 2014. GloVe: Global Vectors for Word Representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, 1532–1543. URL http://www.aclweb.org/anthology/D14-1162.

Salimans, T.; and Kingma, D. P. 2016. Weight normalization: A simple reparameterization to accelerate training of deep neural networks. In *Advances in neural information processing systems*, 901–909.

Shao, T.; Cai, F.; Chen, H.; and de Rijke, M. 2019. Length-Adaptive Neural Network for Answer Selection. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR'19, 869–872. New York, NY, USA: Association for Computing Machinery. ISBN 9781450361729. doi:10.1145/3331184.3331277. URL https://doi.org/10.1145/3331184.3331277.

Shen, D.; Wang, G.; Wang, W.; Min, M. R.; Su, Q.; Zhang, Y.; Li, C.; Henao, R.; and Carin, L. 2018a. Baseline Needs More Love: On Simple Word-Embedding-Based Models and Associated Pooling Mechanisms. *CoRR* abs/1805.09843. URL http://arxiv.org/abs/1805.09843.

Shen, T.; Zhou, T.; Long, G.; Jiang, J.; Pan, S.; and Zhang, C. 2017. Disan: Directional self-attention network for rnn/cnn-free language understanding. *arXiv preprint arXiv:1709.04696* .

Shen, T.; Zhou, T.; Long, G.; Jiang, J.; Wang, S.; and Zhang, C. 2018b. Reinforced Self-Attention Network: a Hybrid of Hard and Soft Attention for Sequence Modeling. *CoRR* abs/1801.10296. URL http://arxiv.org/abs/1801.10296.

Shen, T.; Zhou, T.; Long, G.; Jiang, J.; and Zhang, C. 2018c. Bi-Directional Block Self-Attention for Fast and Memory-Efficient Sequence Modeling. *CoRR* abs/1804.00857. URL http://arxiv.org/abs/1804.00857.

Tenenbaum, M.; and Pollard, H. 1963. *Ordinary differential equations: an elementary textbook for students of mathematics, engineering, and the sciences*. Dover Publications.

Udwadia, F. E.; and Farahani, A. 2008. Accelerated runge-kutta methods. *Discrete Dynamics in Nature and Society* 2008.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, Ł.; and Polosukhin, I. 2017. Attention is all you need. In *Advances in neural information processing systems*, 5998–6008.

Voelker, A.; Kajić, I.; and Eliasmith, C. 2019. Legendre Memory Units: Continuous-Time Representation in Recurrent Neural Networks. In *Advances in Neural Information Processing Systems*, 15570–15579.

Wiebe, J.; Wilson, T.; and Cardie, C. 2005. Annotating Expressions of Opinions and Emotions in Language. *language resources and evaluation* 39: 165–210.

Yang, B.; Li, J.; Wong, D. F.; Chao, L. S.; Wang, X.; and Tu, Z. 2019. Context-aware self-attention networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, 387–394.

Yang, B.; Tu, Z.; Wong, D. F.; Meng, F.; Chao, L. S.; and Zhang, T. 2018. Modeling Localness for Self-Attention Networks. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, 4449–4458. Brussels, Belgium: Association for Computational Linguistics. doi:10.18653/v1/D18-1475. URL https://www.aclweb.org/anthology/D18-1475.

Yang, Y.; Yih, W.-t.; and Meek, C. 2015. WikiQA: A Challenge Dataset for Open-Domain Question Answering. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, 2013–2018. Lisbon, Portugal: Association for Computational Linguistics. doi:10.18653/v1/D15-1237. URL https://www.aclweb.org/anthology/D15-1237.

Yin, W.; Schütze, H.; Xiang, B.; and Zhou, B. 2016. Abcnn: Attention-based convolutional neural network for modeling sentence pairs. *Transactions of the Association for Computational Linguistics* 4: 259–272.

Zhao, J.; Zhan, Z.; Yang, Q.; Zhang, Y.; Hu, C.; Li, Z.; Zhang, L.; and He, Z. 2018. Adaptive learning of local semantic and global structure representations for text classification. In *Proceedings of the 27th International Conference on Computational Linguistics*, 2033–2043.