Multi-Zone Unit for Recurrent Neural Networks

Fandong Meng,[1] **Jinchao Zhang,**[1] **Yang Liu,**[2] **Jie Zhou**[1]

[1]WeChat AI - Pattern Recognition Center Tencent Inc., China
[2]Department of Computer Science and Technology, Tsinghua University, Beijing, China
{fandongmeng, dayerzhang, withtomzhou}@tencent.com
liuyang2011@tsinghua.edu.cn

Abstract

Recurrent neural networks (RNNs) have been widely used to deal with sequence learning problems. The input-dependent transition function, which folds new observations into hidden states to sequentially construct fixed-length representations of arbitrary-length sequences, plays a critical role in RNNs. Based on single space composition, transition functions in existing RNNs often have difficulty in capturing complicated long-range dependencies. In this paper, we introduce a new **M**ulti-**z**one **U**nit (MZU) for RNNs. The key idea is to design a transition function that is capable of modeling multiple space composition. The MZU consists of three components: zone generation, zone composition, and zone aggregation. Experimental results on multiple datasets of the character-level language modeling task and the aspect-based sentiment analysis task demonstrate the superiority of the MZU.

Introduction

Processing sequential data of variable length is a major challenge in the field of natural language processing (NLP). Recurrent Neural Networks (RNNs), Long Short-term Memories (LSTMs) (Hochreiter and Schmidhuber 1997) and Gated Recurrent Units (GRUs) (Cho et al. 2014) in particular, have recently become one of the most popular tools to approach sequence learning tasks, such as handwriting recognition (Graves 2013), sentiment classification (Tang et al. 2016; Wang et al. 2016), sequence labeling (Huang, Xu, and Yu 2015; Akbik, Blythe, and Vollgraf 2018), language modeling (Ha, Dai, and Le 2017; Zoph and Le 2017; Chung, Ahn, and Bengio 2017; Mujika, Meier, and Steger 2017; Li et al. 2018b) and machine translation (Chen et al. 2018; Meng and Zhang 2019).

RNNs sequentially construct fixed-length representations of arbitrary-length sequences by folding new observations into their hidden states using an input-dependent transition operator. The hidden state $\mathbf{h}_t$ is updated recursively using the previous hidden state $\mathbf{h}_{t-1}$ and the current input $\mathbf{x}_t$ as $\mathbf{h}_t = F(\mathbf{h}_{t-1}, \mathbf{x}_t)$, where F is a differentiable function with learnable parameters, such as multiplying its inputs by a matrix and squashing the result with a non-linear function

for a vanilla RNN. The state transition between consecutive hidden states adds a new input to the summary of previous ones. However, this procedure of constructing a new summary from the combination of the previous one and the new input in conventional RNNs only bases on a single space composition, which often has difficulty in capturing complicated long-range dependencies, to allow the hidden state to rapidly adapt to quickly changing modes of the input while still preserving a useful summary of the past.

Recent studies in neural machine translation show that it is beneficial to linearly project one single-space representation into multi-space representations and then capture useful information from these different representation spaces, e.g., the multi-head attention (Vaswani et al. 2017; Chen et al. 2018; Meng and Zhang 2019), which also has been verified effective for capturing multiple semantic aspects from the user utterance in generative dialogue systems (Tao et al. 2018). Although some effective RNN extensions, e.g., GRU and LSTM, have been proposed with gating units controlling the information flow. The gating units are also generated by such kind of function (F) based on single space composition, which is not expressive enough to capture potentially complex information for controlling the information flow.

In this paper, we thus boost the F function through modeling multiple space composition, and propose a **M**ulti-**z**one **U**nit for Recurrent Neural Networks, named MZU. The MZU consists of three components: zone generation, zone composition, and zone aggregation. At each time step, the MZU projects the previous hidden state and the current input into multiple zones (zone generation), then conducts full interactions and compositions on these zones (zone composition), and finally aggregates them to the final representation (zone aggregation). In particular, we propose three effective approaches for the zone composition to verify the MZU. The first model exploits self-attention (Vaswani et al. 2017) to draw global dependencies on zones. The second model utilizes graph convolutional network (Kipf and Welling 2016; 2017) to perform neighborhood mixing on zones by leveraging the graph connectivity structure as a filter. The third model uses the dynamic routing algorithm (Sabour, Frosst, and Hinton 2017) to model part-whole relationships on

zones. To further enhance the MZU, we propose an effective regularization objective to promote the diversity in multiple zones. The architecture of MZU is generic as standard RNNs, thus can be extended to deep MZUs.

We evaluate the MZU on 1) the challenging character-level language modeling task with two standard datasets, namely the well known Penn Treebank (Marcus, Marcinkiewicz, and Santorini 1993) and the larger Wikipedia dataset (text8) (Mahoney 2011); and 2) the aspect-based sentiment analysis (ABSA) task with two datasets of the SemEval 2014 Task 4 (Pontiki et al. 2014) in different domains. Experimental results on both tasks demonstrate the superiority of the MZU to previous competitive RNNs and its generalizability across tasks.

Our main contributions are three-fold:

- We propose a new and generic Multi-zone Unit for RNNs, along with three effective approaches for the zone composition.

- To further enhance the MZU, we propose an effective regularization objective to promote the diversity of multiple zones.

- We provide empirical and visualization analyses to reveal advantages of the MZU, e.g., capturing richer linguistic information and better long sequence processing.

Multi-Zone Unit for RNNs

Overview

MZU maintains a hidden state $\mathbf{h}_t$ to summarize past inputs at each time step t, updated as:

$$\mathbf{h}_t = (1 - \mathbf{g}_t) \odot \mathbf{h}_{t-1} + \mathbf{g}_t \odot \widetilde{\mathbf{h}}_t \quad (1)$$

where $\odot$ is an element-wise product, $\mathbf{g}_t$ is the gate, to control the flow of information in the previous hidden state $\mathbf{h}_{t-1}$ and the candidate activation $\widetilde{\mathbf{h}}_t$ at the current time step, which is computed as:

$$\widetilde{\mathbf{h}}_t = \tanh(M_h(\mathbf{x}_t, \mathbf{h}_{t-1})) \quad (2)$$

where $\mathbf{x}_t$ is the input embedding at time t, and the gate $\mathbf{g}_t$ is computed as:

$$\mathbf{g}_t = \sigma(M_g(\mathbf{x}_t, \mathbf{h}_{t-1})) \quad (3)$$

where M_h and M_g are multi-zone transformation functions, designed for modeling multiple space composition, and their formal definitions will be described in the next section.

MZU extends the standard RNN by replacing the linear transformation (i.e., the F function) on $[\mathbf{x}_t, \mathbf{h}_{t-1}]$ with multi-zone transformation functions, i.e. the M_h and the M_g in Eq.(2) and (3), as well as maintains a simplified gating mechanism to control the information flow. We can also design more complex gating mechanisms as GRU and LSTM to further enhance MZU.

Multi-Zone Transformation Function

The multi-zone transformation function, i.e., the M-function (in Eq. (2) (3)) is designed for enhancing both the transformation module (i.e., the candidate activation $\widetilde{\mathbf{h}}_t$) and the gating module (i.e., the $\mathbf{g}_t$). The M-function only takes the embedding $\mathbf{x}_t$ and the previous hidden state $\mathbf{h}_{t-1}$ as inputs,
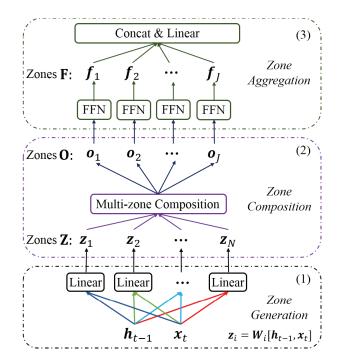

Figure 1: Illustration for the multi-zone transformation function, which is consists of three components. (1) *Zone Generation*: linearly project $[\mathbf{x}_t, \mathbf{h}_{t-1}]$ into N zones $\mathbf{Z}$; (2) *Zone Composition*: conduct compositions on $\mathbf{Z}$ and output J zones $\mathbf{O}$; (3) *Zone Aggregation*: conduct a deep abstraction on $\mathbf{O}$ with a position-wise FFN and output new zones $\mathbf{F}$, then aggregate the representations by concatenating $\mathbf{F}$ and finally conduct a linear transformation to obtain the final representation.

where $\mathbf{x}_t \in \mathbb{R}^{d_x}$ and $\mathbf{h}_{t-1} \in \mathbb{R}^{d_h}$. Figure 1 shows the framework of the M-function, which consists of three components: *Zone Generation*, *Zone Composition*, and *Zone Aggregation*.

Zone Generation.   As in Figure 1, we linearly project the current input and the previous hidden state into N zone vectors $\mathbf{Z} = \{\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_N\}$ parameterized with different weight matrices. Each zone $\mathbf{z}_i$ is computed as:

$$\mathbf{z}_i = \mathbf{W}_i[\mathbf{x}_t, \mathbf{h}_{t-1}] \quad (4)$$

where $\mathbf{W}_i \in \mathbb{R}^{(d_x+d_h) \times d_z}$ (we set $d_z = d_h/N$).

Zone Composition.   Then we conduct a multi-zone composition with fully connected interactions on $\mathbf{Z}$, and output J zones $\mathbf{O} = \{\mathbf{o}_1, \mathbf{o}_2, \ldots, \mathbf{o}_J\}$, where J can be equal to N or not. In this step, zones in $\mathbf{Z}$ can fully interact with each other and generate new representations through our specially designed composition models. This composition can be designed flexibly, in particular, we provide three models, i.e., a self-attention based model, a graph convolutional model and a capsule networks based model, which will be formally described later.

Zone Aggregation. We further conduct a deep abstraction on the newly generated zones $\mathbf{O}$, and aggregate them to the final representation in this step. In particular, we apply a position-wise feed-forward network (FFN) (Vaswani et al. 2017) to each zone $\mathbf{o}_j$ to conduct a deep abstraction and feature extraction, and output to $\mathbf{f}_j$ correspondingly. This consists of two linear transformations with a ReLU activation in between:

$$\mathbf{f}_j = \max(0, \mathbf{o}_j \mathbf{W}_1 + \mathbf{b}_1)\mathbf{W}_2 + \mathbf{b}_2 \qquad (5)$$

where $\mathbf{W}_1 \in \mathbb{R}^{d_o \times d_f}$, and $\mathbf{W}_2 \in \mathbb{R}^{d_f \times d_o}$, d_f is the filter size. The linear transformations are the same across different zones. After that, we obtain J deeply transformed zones $\mathbf{F} = [\mathbf{f}_1, \mathbf{f}_2, ..., \mathbf{f}_J]^\top$. Then, we aggregate the representations by concatenating these zones and conduct a linear transformation to obtain the final representation.

Models for Zone Composition

To be noticed that, how to conduct full interactions on the multiple zones $\mathbf{Z}$ and generate new zones $\mathbf{O}$, i.e., $\mathbf{Z} \to \mathbf{O}$ in the "*Zone Composition*" (Figure 1-(2)) is crucial. It can be flexibly designed, in this paper, we just propose three effective approaches:

1) We apply self-attention (Vaswani et al. 2017) to compute the relevance between each zone pair in $\mathbf{Z}$, and generate new zones $\mathbf{O}$ by a weighted sum of linear transformed $\mathbf{Z}$, since self-attention has been proved effective to draw global dependencies of representations on different positions.

2) We exploit Graph Convolutional Network (GCN) (Kipf and Welling 2016; 2017) to conduct interactions on zones in $\mathbf{Z}$ by taking the zones as nodes of a graph, and generate new zones $\mathbf{O}$ by absorbing information from their neighbor nodes along weighted edges. As a special form of Laplacian Smoothing (Li, Han, and Wu 2018), GCN takes the graph connectivity structure as a filter to perform neighborhood mixing, and therefore better for zone composition.

3) We utilize the capsule networks (Hinton, Krizhevsky, and Wang 2011; Sabour, Frosst, and Hinton 2017; Hinton, Sabour, and Frosst 2018), which have strong capabilities in representation composition and have been successfully applied to some NLP tasks (Zhao et al. 2018; Wang et al. 2018; Zhang et al. 2019). In particular, we model part-whole relationships between the older zones $\mathbf{Z}$ and the new zones $\mathbf{O}$ to aggregate diverse and useful information from older ones $\mathbf{Z}$, and achieve better representation composition through iterations of dynamic routing (Sabour, Frosst, and Hinton 2017).

Self-attention based MZU (satMZU) We firstly project the N zones into N queries, N keys and N values of dimension d_z, each with a linear transformation, parameterized with different weights. Then we compute the dot products of the query with all keys, divide each by $\sqrt{d_z}$, and apply a softmax function to obtain the weights on the values. In practice, we compute the attention function on a set of queries simultaneously, packed together into a matrix $\mathbf{Q}$. The keys and values are also packed together into matrices $\mathbf{K}$ and $\mathbf{O}$. Then the matrix of outputs is computed as:

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{O}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d_z}}\right)\mathbf{O}, \qquad (6)$$

which is the N new zones $\mathbf{O} = [\mathbf{o}_1, \mathbf{o}_2, ..., \mathbf{o}_N]^\top$.

Graph Convolutional MZU (gcnMZU) We take the N zones as N nodes and construct a graph $G = (V, E)$, where V and E represent the sets of nodes and edges, with $|V| = N$ and $|E| = N * N$, respectively. Given node $\mathbf{z}_i$ and node $\mathbf{z}_j$, we use cosine to compute the semantic relevance between two nodes, as the weighted edge w_{ij}, which is calculated by:

$$w_{ij} = \frac{\mathbf{z}_i^\top \mathbf{z}_j}{\|\mathbf{z}_i\| \cdot \|\mathbf{z}_j\|} \qquad (7)$$

where $\|\cdot\|$ stands for the L2 norm. Then we obtain the adjacent matrix[1] $\tilde{A} \in \mathbb{R}^{|V| \times |V|}$:

$$\tilde{A} = \begin{bmatrix} w_{11} & w_{12} & ... & w_{1N} \\ ... & ... & ... & ... \\ w_{N1} & w_{N2} & ... & w_{NN} \end{bmatrix} \qquad (8)$$

The degree matrix is represented as $\tilde{D}$, where $\tilde{D}_{ii} = \sum_j \tilde{A}_{ij}$. Meanwhile, we concatenate the representation of each node to a matrix $\mathbf{Z} = [\mathbf{z}_1, \mathbf{z}_2, ..., \mathbf{z}_N]^\top \in \mathbb{R}^{|V| \times d_z}$. Then we obtain the new node matrix $\mathbf{O}$ through a layer of convolution, which has incorporated the structured interactive information, as follows:

$$\mathbf{O} = \sigma\left(\tilde{D}^{-\frac{1}{2}} \tilde{A} \tilde{D}^{-\frac{1}{2}} \mathbf{Z} \mathbf{W}_g\right) \qquad (9)$$

where $\mathbf{W}_g \in \mathbb{R}^{|V| \times d_g}$ is the weight matrix, and d_g (we set $d_g = d_z$) is the dimension of this convolutional layer, and $\mathbf{O} \in \mathbb{R}^{|V| \times d_g}$. It is a special form of Laplacian Smoothing (Li, Han, and Wu 2018), which computes the new representation of a node as the weighted average of itself and its neighbors'.

Capsule Networks based MZU (capMZU) We take the N zones $\mathbf{Z} = \{\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_N\}$ as N low-level capsules, and aim to output J high-evel capsules (i.e., zones) $\mathbf{O} = \{\mathbf{o}_1, \ldots, \mathbf{o}_J\}$, where $\mathbf{o}_j \in \mathbb{R}^{d_o}$ (we set $d_o = d_h/J$). We aggregate information from $\mathbf{Z}$ and conduct representation composition to generate $\mathbf{O}$, by modeling part-whole relationships between capsules in $\mathbf{Z}$ and capsules in $\mathbf{O}$.

We firstly initialize the initial logits b_{ij} with 0, which are log prior probabilities that capsule $\mathbf{z}_i$ should be coupled to capsule $\mathbf{o}_j$. And we generate the "prediction vectors" $\hat{\mathbf{z}}_{j|i}$ from the low-level capsules as follows:

$$\hat{\mathbf{z}}_{j|i} = \mathbf{z}_i \mathbf{W}_{ij}^c \qquad (10)$$

where $\mathbf{W}^c \in \mathbb{R}^{N \times J \times d_z \times d_o}$ is the weight tensor.[2]

Then we aggregate information from $\mathbf{Z}$ and conduct representation composition to generate $\mathbf{O}$, through T iterations of dynamic routing on $\hat{\mathbf{z}}_{j|i}$. At each iteration, we firstly compute coupling coefficient c_{ij} between low-level capsule $\mathbf{z}_i$ and all high-level capsules, by a "softmax" on initial logits b_{ij}, as follows:

$$c_{ij} = \text{softmax}(\mathbf{b}_i) = \frac{\exp(b_{ij})}{\sum_k \exp(b_{ik})} \qquad (11)$$

---

[1]Self-connections have been added after the computation.

[2]In practice, we share the transformation matrix of each output capsule o_j among all the input capsules, that is $\mathbf{W}^c \in \mathbb{R}^{J \times d_z \times d_o}$.

After that we generate $\mathbf{s}_j$ by a weighted sum over $\hat{\mathbf{z}}_{j|i}$ with coupling coefficient c_{ij} as weights, computed as:

$$\mathbf{s}_j = \sum_i c_{ij}\hat{\mathbf{z}}_{j|i} \tag{12}$$

Then we take $\mathbf{s}_j$ as input and output capsule $\mathbf{o}_j$ by squashing $\mathbf{s}_j$ as follows:

$$\mathbf{o}_j = \text{squash}(\mathbf{s}_j) = \frac{||\mathbf{s}_j||^2}{1+||\mathbf{s}_j||^2}\frac{\mathbf{s}_j}{||\mathbf{s}_j||} \tag{13}$$

where $\text{squash}(\cdot)$ (Sabour, Frosst, and Hinton 2017) is a non-linear function to ensure that short vectors get shrunk to almost zero length and long vectors get shrunk to a length slightly below 1.

Then b_{ij} are iteratively refined by measuring the agreement (scalar product) between the current output $\mathbf{o}_j$ and the prediction $\hat{\mathbf{z}}_{j|i}$:

$$b_{ij} = b_{ij} + \hat{\mathbf{z}}_{j|i} \cdot \mathbf{o}_j \tag{14}$$

After T iterations of dynamic routing, we obtain J final capsules (zones) $\mathbf{O} = [\mathbf{o}_1, \mathbf{o}_2, ..., \mathbf{o}_J]^\top$.

Deep Transition MZU

The MZU is generic as standard RNNs, and can be extended to any deep MZUs. Since recent studies (Pascanu et al. 2014; Barone et al. 2017; Meng and Zhang 2019) have demonstrated the superiority of deep transition RNNs over deep stacked RNNs, we extend the MZU to a more powerful deep transition MZU. The entire deep transition block is composed of one MZU cell followed by several transition MZU (t-MZU) cells at each time step. As a special case of MZU, t-MZU only has "state" as input, i.e. the embedding input of t-MZU is a zero vector. In the whole recurrent procedure, for the current time step, the "state" output of one MZU/t-MZU cell is used as the "state" input of the next t-MZU cell. And the "state" output of the last t-MZU cell for the current time step is carried over as the "state" input of the first MZU cell for the next time step.

For a t-MZU cell, each hidden state at time step t at transition depth l is computed as follows:

$$\mathbf{h}_t^l = (1-\mathbf{g}_t^l) \odot \mathbf{h}_t^{l-1} + \mathbf{g}_t^l \odot \widetilde{\mathbf{h}}_t^l \tag{15}$$

$$\widetilde{\mathbf{h}}_t^l = \tanh(M_h^l(0, \mathbf{h}_t^{l-1})) \tag{16}$$

where the gate $\mathbf{g}_t^l$ is computed as:

$$\mathbf{g}_t^l = \sigma(M_g^l(0, \mathbf{h}_t^{l-1})) \tag{17}$$

where M_h^l and M_g^l are Multi-Zone Transformation functions, as described in Section .

Regularization on Multiple Zones

We expect that the representations in multiple zones of MZU are as different as possible. Thus we propose a disagreement regularization on these zones, inspired by Li et al. (2018a). This regularization is designed to maximize the cosine distance (i.e., negative cosine similarity) between zone pairs. Our objective is to enlarge the average cosine distance among all zone pairs, computed as:

$$D_{\text{zone}} = -\frac{1}{N^2}\sum_{i=1}^N\sum_{j=1}^N\frac{\mathbf{z}_i^\top \mathbf{z}_j}{||\mathbf{z}_i|| \cdot ||\mathbf{z}_j||} \tag{18}$$

Training Objective. Taking into the regularization, the training objective of MZU is:

$$J(\theta) = \arg\max_\theta \left\{ \underbrace{L(\theta)}_{\text{likelihood}} + \lambda * \underbrace{\sum_{s}^{s}\sum_{m}^{m} D_{\text{zone}}(\theta)}_{\text{disagreement}} \right\} \tag{19}$$

where m is the number of M-functions in the MZU, s is the number of inputs (e.g., tokens), $L(\theta)$ is a task specific training objective, and λ is a hyper-parameter used to balance the preference between two terms of the loss function.

Experiments

We verify the MZU on the character-level language modeling task and the aspect based sentiment analysis task.

Character-level Language Modeling

We use two standard datasets, namely the Penn Treebank (Marcus, Marcinkiewicz, and Santorini 1993) and the larger Wikipedia dataset (text8) (Mahoney 2011).

Penn Treebank. The Penn Treebank dataset is a collection of Wall Street Journal articles written in English. We follow the process procedure introduced in (Mikolov et al. 2012), and split the data into training, validation and test sets consisting of 5.0M, 390K and 440K characters, respectively.

The Wikipedia Corpus (text8). The text8 dataset consists of 100M characters extracted from the English Wikipedia. text8 contains only alphabets and spaces, and thus we have total 27 symbols. In order to compare with other previous works, we follow the the process procedure in (Mikolov et al. 2012), and split the data into training, validation and test sets consisting of 90M, 5M, and 5M characters, respectively.

Training Details. We train the model using Adam (Kingma and Ba 2014) with an initial learning rate of 0.001. Each update is done by using a mini-batch of 256 examples. We use the truncated backpropagation through time to approximate the gradients, setting the length to 150. The norm of the gradient is clipped with 5.0. We apply layer normalization to our models, and apply dropout to the candidate activation to avoid overfitting. The dropout rate for the Penn Treebank task and the text8 task are set to 0.5 and 0.3, respectively. The hidden size and filter size for Penn Treebank are set to 800 and 1000, respectively. And those for text8 are set to 1,536 and 3,072, respectively. The embedding size is 256. We share parameters of the M-function across different depths (i.e., the deep transition counterpart) for Penn Treebank. We measure the performance of checkpoints by evaluating *bits per character* (BPC) over the valid set, and make the test with the best one. BPC is the negative log-likelihood divided by natural logarithm of 2. The lower the better.

System Description. We use satMZU, gcnMZU and capMZU standing for the self-attention based MZU, the graph convolutional MZU and the capsule networks based MZU, respectively. "DT" stands for deep transition MZU

MODEL	SIZE	BPC
LSTM (Krueger et al. 2017)	–	1.36
Zoneout LSTM (Krueger et al. 2017)	–	1.27
2-Layers LSTM (Zoph and Le 2017)	6.6M	1.243
LN-HM-LSTM (Chung, Ahn, and Bengio 2017)	–	1.24
HyperLSTM (Ha, Dai, and Le 2017)	14.4M	1.219
NASCell (Zoph and Le 2017)	16.3M	1.214
FS-LSTM-4 (Mujika, Meier, and Steger 2017)	7.2M	1.190
res-IndRNN (Li et al. 2018b)	–	1.19
satMZU+DT (ours)	6.5M	1.189
gcnMZU+DT (ours)	6.4M	1.183
capMZU+DT (ours)	7.2M	**1.181**

Table 1: BPC on Penn Treebank test set.

MODEL	SIZE	BPC
Large td-LSTM (Zhang et al. 2016)	–	1.49
MI-LSTM (Wu et al. 2016)	–	1.44
BN-LSTM (Cooijmans et al. 2017)	–	1.36
Zoneout LSTM (Krueger et al. 2017)	–	1.336
LN HM-LSTM (Chung, Ahn, and Bengio 2017)	35M	1.29
Large RHN (Zilly et al. 2017)	45M	1.27
Large mLSTM (Krause et al. 2016)	45M	1.27
satMZU+DT (ours)	32M	1.268
gcnMZU+DT (ours)	32M	1.260
capMZU+DT (ours)	41M	**1.249**
Transformer (Al-Rfou et al. 2018)	235M	1.13
Transformer-XL (Dai et al. 2019)	277M	1.08

Table 2: BPC on text8 test set.

#ZONES	satMZU	gcnMZU	capMZU
2	1.191	1.187	1.186
4	**1.189**	**1.183**	**1.181**
8	1.200	1.189	**1.181**
16	1.208	1.185	1.186

Table 3: Effects of zone numbers on Penn Treebank test set. All the tested models are with "DT", which omitted due to the limited table space.

MODEL	NO-DT	DT
GRU	1.269	1.208
satMZRNN	**1.214**	**1.189**
REGULAR-GATE	1.236	1.207
REGULAR-TRANS.	1.247	1.208
gcnMZRNN	**1.196**	**1.183**
REGULAR-GATE	1.224	1.197
REGULAR-TRANS.	1.225	1.191
capMZRNN	**1.192**	**1.181**
REGULAR-GATE	1.219	1.190
REGULAR-TRANS.	1.211	1.188

Table 4: Ablation study of M-functions in MZUs. "REGULAR" stands for replacing the M-function with the regular linear transformation function on $[\mathbf{x}_t, \mathbf{h}_{t-1}]$.

cells, with transition depth setting to 1 by default, i.e. one MZU cell plus one t-MZU cell. For all MZU models, we apply 4 zones in M-functions, except when otherwise mentioned. For all capMZUs, we set the number of output capsules to 2, and use 3 routing iterations, according to primary experiments. We set λ to 1.0 in Eq. 19 for all experiments. From primary experiments with different settings to λ, we find setting $\lambda > 0$ (promoting to different zones) consistently improve the performance ($0.005 \sim 0.007$ BPC) on both shallow and deep MZU models. While setting $\lambda < 0$ (promoting to similar zones), e.g. -0.5 or -1.0, sharply declines the performance ($0.03 \sim 0.05$ BPC). This indicates modeling multi-zone transformation affects the state transition, and consequently affects the performance.

Main Results. Table 1 shows results on Penn Treebank test set, with comparing to existing competitive approaches. Our MZU models outperform previous competitive approaches, including deeper RNNs, e.g., the 4-layers FS-LSTM and the 11-layers res-IndRNN, without using more parameters. Among three MZU models, the gcnMZU+DT and the capMZU+DT achieve better performance than the satMZU+DT, and the capMZU+DT achieves the best BPC with 1.181. The capMZU can model part-whole relationships between the old zones and the new zones to aggregate useful information, through multiple iterations of routing-by-agreement. Such kind of multi-round refinements let the new zones contain more accurate representations and therefore capMZU preforms better than other MZUs.

To demonstrate that MZUs work well across datasets, we conduct experiments on a larger dataset text8. As shown

in Table 2, using even less parameters, our satMZU+DT, gcnMZU+DT and capMZU+DT outperform several previous deep RNNs, including the 10-depths Large RHN and the stacked Large mLSTM, except the recently proposed transformer-based models (Al-Rfou et al. 2018; Dai et al. 2019) trained with much longer context (512), deeper layers (24-layers Transformer & 64-layers Transformer-XL) and larger amount of parameters. Explicit using longer context indeed improves the performance of language modeling. While in this work, we mainly focus on boosting the transition functions (i.e., only operate on $\mathbf{x}_t$ and $\mathbf{h}_{t-1}$) of RNNs. Our approaches can also be applied on transformer-based architectures, e.g for enhancing the multi-head attention.

The Number of Zones. We investigate the impact of zone numbers on MZUs, in particular, we conduct experiments with three deep MZU models (MZU+DT) on Penn Treebank. For each model, we respectively test 2, 4, 8, and 16 zones in M-functions. As the dimension of the zone is decided by the hidden size and the zone number, i.e., $d_z = d_h/N$, more zones do not lead to more parameters. As shown in Table 3, using 4 zones achieves the best performance for all MZU models. Additionally, we observe that the satMZU is more sensitive on the number of zones, and leveraging too many zones (e.g. 8, 16) sharply declines the performance. The gcnMZU and the capMZU are more robust on modeling multiple zones, benefiting from the powerful graph convolutional structure and dynamic routing algorithm, respectively.
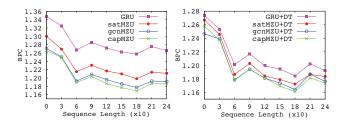
Figure 2: BPC of shallow (left) and deep (right) models on sequences with different lengths. The numbers on X-axis stand for sequences longer than the corresponding length (x10), e.g., 9 for sequences with (90, 120] characters. The superiority of MZUs over GRUs is more obvious on long sequences (e.g. >90).

Ablation Study. Since the M-function works on the gating module (i.e., g_t) and the transformation module (i.e., $\widetilde{h}_t$), we conduct an ablation study to investigate its effects on both modules. We show results in Table 4, where "REGULAR" stands for replacing the M-function with the regular linear transformation on $[x_t, h_{t-1}]$. As shown in Table 4, the M-function is crucial for both the gating module ("GATE") and the transformation module ("TRANS."), for both shallow ("NO-DT") and deep ("DT") MZU models, since replacing any module with the REGULAR one sharply declines the performance. We also list results[3] of GRU and deep transition GRU (Pascanu et al. 2014) (i.e., "DT") in Table 4. Even with the module degeneration, our MZUs still outperform GRUs which only based on single space composition, in shallow and deep models, correspondingly.

About Length. We investigate effects of MZU models on sequences with different length, and compare them with GRU and deep transition GRU (GRU+DT) in Figure 2. The results of shallow and deep models are in the left and right subfigure, respectively. As shown in Figure 2, on sequences with different length, shallow and deep MZU models consistently yield better performance than the GRU and the GRU+DT, respectively. And the superiority is more obvious on long sequences (e.g., >90) for both shallow and deep MZU models, demonstrating the stronger capability for long sequence processing of the MZU.

Visualization Analysis. We conduct a visualization analysis to demonstrate the superiority of the MZU (Figure 3(b)) over GRU (Figure 3(a)). In particular, we compare the capMZU to GRU. We compute the relevance between each of the last five characters (i.e., 'm', 'e', 'n', 't', 's') and all characters before it. For example, the relevance between 'm' and the first character 'p' is calculated by the *cosine* similarity between the candidate activation (stands for the transformation at current step) of 'm' and the hidden state of 'p'. The higher the relevance, the more white of the grayscale.

---

[3]We also apply dropout and layer normalization to GRUs, and build models with the same settings to our MZUs.
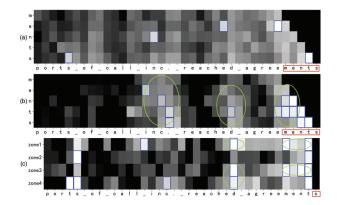

Figure 3: Relevance results between each of the last five characters (i.e., 'm', 'e', 'n', 't', 's') and all characters before it, from the GRU(a) and the capMZU(b). (c) shows the zone-specific relevance result of capMZU between the last character 's' and all characters before it. The higher the relevance, the more white of the grayscale. Blue rectangles and green ellipses indicate high-relevance parts. '_' stands for the space between two words.

SENTENCE	The appetizers are ok, but the service is slow.	
Aspect	The appetizers	service
Sentiment	Neutral	Negative

Table 5: An example that contains different sentiment polarities towards two aspects.

From Figure 3(a) (GRU), we can find that 1) The gray distribution is not sharply contoured; and 2) the highest relevance positions to each computed character usually occur at the nearest position to it or at positions with the same surface character, such as the boxes marked blue. In Figure 3(b) (capMZU), the gray distribution is more sharply contoured, e.g., three regions marked by green ellipses. Additionally, it captures some philological segments, such as the suffixes "ments" and "ed", and parts of the word "inc.", which are related to the word "agreements". Furthermore, we investigate how multiple zones affect the performance, and show the zone-specific relevance between the last character 's' and all characters before it in Figure 3(c) (capMZU). We find that different zones indeed capture different philological information. For example, "zone1" and "zone3" capture some suffixes (e.g., "ment", "ed"), "zone2" captures the same surface character 's' in long distance, and "zone4" not only captures the nearest 't' to 's', but also captures the long distance and frequent collocation "ts".

Aspect Based Sentiment Analysis

Task Description. Since RNNs are widely used for sequence modeling and classification tasks, we choose the well-known and challenging ABSA task as a case study to demonstrate the generalizability across tasks of the MZU. We test on the aspect-term based task of the SemEval 2014 Task 4 (Pontiki et al. 2014) with two datasets that contain domain-specific customer reviews for restaurants and lap-

MODELS	RESTAURANT		LAPTOP	
	DS	HDS	DS	HDS
TD-LSTM* (Tang et al. 2016)	73.44±1.17	56.48±2.46	62.23±0.92	46.11±1.89
ATAE-LSTM* (Wang et al. 2016)	73.74±3.01	50.98±2.27	64.38±4.52	40.39±1.30
GCAE* (Xue and Li 2018)	77.28±0.32	56.73±0.56	69.14±0.32	47.06±2.45
GRU (WITH-DT)	78.25±0.51	56.17±0.72	71.01±0.73	47.03±1.15
*cap*MZRNN (WITH-DT)	**79.40±0.23**	**57.59±0.70**	**72.45±0.68**	**47.96±1.12**

Table 6: The accuracy (mean of five times training with standard deviation) on SemEval 2014 term-based datasets. Results (marked with "*") of existing systems are cited from (Xue and Li 2018).

tops, each of which contains four sentiment labels (i.e., Positive, Negative, Neutral, and Conflict). For this task, the goal is to infer sentiment polarity over the aspect (i.e., the given term), which is a subsequence of the sentence. The example in Table 5 shows the customer's different attitudes towards two terms: "*The appetizers*" and "*service*". For each dataset, we follow (Xue and Li 2018; Liang et al. 2019) and construct a hard sub-dataset (named "HDS") by extracting from the full dataset (named "DS"). In "HDS", all sentences contain multiple aspects, each of which corresponds to a dfferent sentiment label. On this sub-dataset, we copy each sentence n times, where n is the number of aspects in a sentence.

Models. We compare our MZU with the deep transition GRU and several previous competitive systems. For the MZU, we use *cap*MZRNN with deep transition. We take the ABSA as a sentence-level classification task. Therefore, for both GRU model and *cap*MZRNN model, we follow (Tang et al. 2016), and firstly encode the sentence as a standard Bi-RNN encoder does. Then we conduct mean pooling on the output hidden states to generate a sentence vector. Next we concatenate the sentence vector with the given aspect embedding (the mean of embeddings for multi-word aspects). Finally, we predict the sentiment polarity with the concatenated representation by a softmax layer.

Results. As shown in Table 6, our implemented deep transition GRU outperforms most previous LSTM-based systems (i.e., TD-LSTM (Tang et al. 2016), ATAE-LSTM (Wang et al. 2016) and the gated CNN-based system GCAE (Xue and Li 2018) (comparable on "HDS" sub-datasets), demonstrating that our deep transition GRU is a strong baseline. When incorporating the MZU, our *cap*MZRNN further improve the performance on multiple datasets by significant margins. These results demonstrate the superiority and generalizability of the MZU cross tasks.

Related Work

This work is inspired by the idea of processing attentions on multiple subspaces with different heads (Vaswani et al. 2017). While different from this multi-channel style operation, we conduct fully connected interactions on multiple zones to extract useful information. In particular, we propose three effective models to conduct interactions and representation composition.

Some researchers (Pascanu et al. 2014; Zilly et al. 2017; Graves 2016; Mujika, Meier, and Steger 2017) boost the

transition function with deep architectures, such as adding intermediate layers to increase the transition depth. Different from these works, we boost the performance of RNNs by modeling multiple zone composition for the transition function. Additionally, our MZU is generic and can be easily extended to deep MZUs.

Some researchers devise models to capture specific features from the processed history sequence, including incorporating an attention mechanism into LSTM cells (Zhong, Lin, and Chen 2018) or extending the GRU with a recurrent attention unit (Zhong, Yue, and Ling 2018). Different from these works, we focus on boosting the transition function by modeling multiple zone composition, which only operates on $\mathbf{x}_t$ and $\mathbf{h}_{t-1}$, at each time step.

Some studies (Merity, Keskar, and Socher 2018; Brahma 2018) investigate regularization techniques for optimizing LSTM-based models, such as weight dropout, variational dropout, weight tying, temporal activation regularization, and past decode regularization. These techniques can be used to further optimize MZU-based models in the future.

Conclusion

We propose a generic Multi-zone Unit for RNNs along with three effective variants (i.e., the *sat*MZU, the *gcn*MZU and the *cap*MZU), which boost the state transition function through modeling multiple space composition. To further enhance MZUs, we propose an effective regularization objective to promote the diversity in multiple zones. Experiments on two character-level language modeling datasets show that our MZUs can automatically capture rich linguistic information and substantially improve the performance. Experimental results on the ABSA task further demonstrate the superiority and generalizability of the MZU.

References

Akbik, A.; Blythe, D.; and Vollgraf, R. 2018. Contextual string embeddings for sequence labeling. In *ACL*.

Al-Rfou, R.; Choe, D.; Constant, N.; et al. 2018. Character-level language modeling with deeper self-attention. *arXiv preprint arXiv:1808.04444*.

Barone, A. V. M.; Helcl, J.; Sennrich, R.; et al. 2017. Deep architectures for neural machine translation. In *WMT*.

Brahma, S. 2018. Improved language modeling by decoding the past. *CoRR abs/1808.05908*.

Chen, M. X.; Firat, O.; Bapna, A.; et al. 2018. The best of both worlds: Combining recent advances in neural machine translation. In *ACL*.

Cho, K.; van Merrienboer, B.; Gulcehre, C.; et al. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. In *EMNLP*.

Chung, J.; Ahn, S.; and Bengio, Y. 2017. Hierarchical multiscale recurrent neural networks. In *ICLR*.

Cooijmans, T.; Ballas, N.; Laurent, C.; et al. 2017. Recurrent batch normalization. In *ICLR*.

Dai, Z.; Yang, Z.; Yang, Y.; et al. 2019. Transformer-xl: Attentive language models beyond a fixed-length context. *arXiv preprint arXiv:1901.02860*.

Graves, A. 2013. Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*.

Graves, A. 2016. Adaptive computation time for recurrent neural networks. *CoRR abs/1603.08983*.

Ha, D.; Dai, A.; and Le, Q. V. 2017. Hypernetworks. In *ICLR*.

Hinton, G. E.; Krizhevsky, A.; and Wang, S. D. 2011. Transforming auto-encoders. In *ICANN*.

Hinton, G. E.; Sabour, S.; and Frosst, N. 2018. Matrix capsules with em routing. In *ICLR*.

Hochreiter, S., and Schmidhuber, J. 1997. Long short-term memory. *Neural computation*.

Huang, Z.; Xu, W.; and Yu, K. 2015. Bidirectional lstm-crf models for sequence tagging. *arXiv*.

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

Kipf, T. N., and Welling, M. 2016. Variational graph auto-encoders. *arXiv preprint arXiv:1611.07308*.

Kipf, T. N., and Welling, M. 2017. Semi-supervised classification with graph convolutional networks. In *ICLR*.

Krause, B.; Lu, L.; Murray, I.; et al. 2016. Multiplicative lstm for sequence modelling. In *ICLR (Workshop)*.

Krueger, D.; Maharaj, T.; Kramár, J.; et al. 2017. Zoneout: Regularizing rnns by randomly preserving hidden activations. In *ICLR*.

Li, J.; Tu, Z.; Yang, B.; et al. 2018a. Multi-head attention with disagreement regularization. In *EMNLP*.

Li, S.; Li, W.; Cook, C.; et al. 2018b. Independently recurrent neural network (indrnn): Building a longer and deeper rnn. In *CVPR*.

Li, Q.; Han, Z.; and Wu, X.-M. 2018. Deeper insights into graph convolutional networks for semi-supervised learning. In *AAAI*.

Liang, Y.; Meng, F.; Zhang, J.; Xu, J.; Chen, Y.; and Zhou, J. 2019. A novel aspect-guided deep transition model for aspect based sentiment analysis. In *EMNLP*.

Mahoney, M. 2011. Large text compression benchmark.

Marcus, M. P.; Marcinkiewicz, M. A.; and Santorini, B. 1993. Building a large annotated corpus of english: The penn treebank. *Computational linguistics* 19(2).

Meng, F., and Zhang, J. 2019. DTMT: A novel deep transition architecture for neural machine translation. *AAAI*.

Merity, S.; Keskar, N. S.; and Socher, R. 2018. Regularizing and optimizing lstm language models. In *ICLR*.

Mikolov, T.; Sutskever, I.; Deoras, A.; et al. 2012. Subword language modeling with neural networks.

Mujika, A.; Meier, F.; and Steger, A. 2017. Fast-slow recurrent neural networks. In *NIPS*.

Pascanu, R.; Gulcehre, C.; Cho, K.; et al. 2014. How to construct deep recurrent neural networks. In *ICLR*.

Pontiki, M.; Galanis, D.; Pavlopoulos, J.; et al. 2014. Semeval-2014 task 4: Aspect based sentiment analysis. In *SemEval*.

Sabour, S.; Frosst, N.; and Hinton, G. E. 2017. Dynamic routing between capsules. In *NIPS*.

Tang, D.; Qin, B.; Feng, X.; et al. 2016. Effective lstms for target-dependent sentiment classification. In *COLING*.

Tao, C.; Gao, S.; Shang, M.; et al. 2018. Get the point of my utterance! learning towards effective responses with multi-head attention mechanism. In *IJCAI*.

Vaswani, A.; Shazeer, N.; Parmar, N.; et al. 2017. Attention is all you need. In *NIPS*.

Wang, Y.; Huang, M.; zhu, x.; et al. 2016. Attention-based lstm for aspect-level sentiment classification. In *EMNLP*.

Wang, Q.; Qiu, J.; Zhou, Y.; et al. 2018. Recurrent capsule network for relations extraction: A practical application to the severity classification of coronary artery disease. *CoRR abs/1807.06718*.

Wu, Y.; Zhang, S.; Zhang, Y.; et al. 2016. On multiplicative integration with recurrent neural networks. In *NIPS*.

Xue, W., and Li, T. 2018. Aspect based sentiment analysis with gated convolutional networks. In *ACL*.

Zhang, S.; Wu, Y.; Che, T.; et al. 2016. Architectural complexity measures of recurrent neural networks. In *NIPS*.

Zhang, X.; Li, P.; Jia, W.; and Zhao, H. 2019. Multi-labeled relation extraction with attentive capsule network. In *AAAI*.

Zhao, W.; Ye, J.; Yang, M.; et al. 2018. Investigating capsule networks with dynamic routing for text classification. In *EMNLP*.

Zhong, G.; Lin, X.; and Chen, K. 2018. Long short-term attention. *CoRR abs/1810.12752*.

Zhong, G.; Yue, G.; and Ling, X. 2018. Recurrent attention unit. *CoRR abs/1810.12754*.

Zilly, J. G.; Srivastava, R. K.; Koutník, J.; et al. 2017. Recurrent highway networks. In *ICML*.

Zoph, B., and Le, Q. V. 2017. Neural architecture search with reinforcement learning. In *ICLR*.