# Sliding Hierarchical Recurrent Neural Networks for Sequence Classification

Bo Li* §, Zhonghao Sheng† §, Wei Ye* ¶, Jinglei Zhang*, Kai Liu‡ and Shikun Zhang*,
*National Engineering Research Center for Software Engineering, Peking University
†School of Software and Microelectronics, Peking University
‡School of Computing, Clemson University
Email: deepblue.lb@gmail.com,
{zhonghao.sheng, wye, jinglei.zhang}@pku.edu.cn, kail@clemson.edu, zhangsk@pku.edu.cn

***Abstract*—Hierarchical Recurrent Neural Networks (HRNN) is an important advance in improving efficiency and performance of sequence classification in recent years. The intuition behind this approach is to slice long sequences into many short sub-sequences and process them in parallel, then capturing the long-term dependencies between those sub-sequences by deeper layers of the networks. In this paper, we propose a novel architecture called Sliding Hierarchical Recurrent Neural Network (SHRNN). We introduce a new sliding mechanism on the input sequence of each layer, named recursive block, so that SHRNN can process the input sequence effectively. We also introduce layer-wise attention and multi-layer regularization for further improvements. We perform large-scale experiments in sequence classification task of both text and image on 8 datasets. As result, we not only achieve new start-of-the-art performance on all datasets by SHRNN, but also investigate effects of different components of SHRNN systematically and thoroughly, which provides best practice for the usage of SHRNN.**

## I. Introduction

Sequence classification is one of the fundamental tasks in natural language processing (NLP) , that is, given a sequence, the task is aiming to assign a class label to this sequence [1]. It has board applications including text classification [2], sentiment analysis [3] and natural language inference [4], [5]. Most researchers use recurrent neural networks (RNNs) to solve the above problems because RNNs have the ability to capture the order information from sequential data, building connection between current input and its context. However, due to the recurrent structure, RNNs suffer from vanishing (and exploding) gradient problem, and lack of ability to capture the long-term dependencies [6], leading effectiveness problem. Besides, RNNs are difficult to be processed in parallel, resulting in the training efficiency problem.

Researchers have done lots of efforts to solve the drawbacks of RNNs. On the one hand, some works try to increase the speed of RNNs by simplifying the recursive unit, such as minimal gated unit (MGU) [7] and simple recurrent unit (SRU) [8]. These models can achieve much higher speed than standard RNNs with slightly improved results. On the other hand, a lot of previous work has been done to improve the ability of capturing long-term dependencies. Gate mechanism is used in Long short-term memory networks (LSTMs) [9] and gated recurrent units (GRUs) [10]. Researchers have also proposed various models based on tree-LSTM [11], [12], skip-connection [13], [14], auxiliary loss [15], reinforcement learning [16]. However, these improved models are often much more time-consuming than standard RNNs, and do not always have a remarkable improvement in results.

Building Hierarchical RNN (HRNN) is an important advance in improving efficiency and performance of sequence classification [17]–[19]. The intuition behind this approach involves two aspects. The first is to divide the long sequence into many short sub-sequences, which can be processed in parallel to improve efficiency. And the other one is to capture the long-term dependencies between those sub-sequences by deeper layers of the neural networks. Representation with increasing levels of abstraction in deeper layers can usually improve the overall performance. There are some varieties of HRNN proposed in recent year. For example, Hierarchical RNN Autoencoder [20] shows that hierarchical RNNs can obtain a better sequence representation. Hierarchical attention networks (HAN) [2] divides the long sequence by sentences and paragraphs to realize a hierarchical recurrent neural network of 2-layers with attention mechanism. Sliced Recurrent Neural Networks (SRNN) [21] and BPIE-BiSRNN [22] slices the input sequence into many fixed length sub-sequences to realize parallel processing.

In this paper, we propose a novel architecture called Sliding Hierarchical Recurrent Neural Network (SHRNN) to better parse sequential data. We introduce a new sliding mechanism and apply layer-wise attention and multi-layer regularization to process the input sequence in a more effective way. Meanwhile, we want to investigate best practices of how to use these components in SHRNN. In a sense, this is similar to how to construct efficient convolutional Neural Networks (CNN) using convolutional layer and pooling layer of various sizes and types.

Just like each element of a sequence is processed by a recursive unit in a vanilla RNN, in SHRNN each sub-sequence can be viewed as being processed by a recursive block, which is the fundamental concept of SHRNN. We note that temporal convolutional network (TCN) [23] employs a similar approach ,which captures long-term dependencies by enlarging

§Both authors contributed equally to this research.
¶Corresponding author.

978-1-7281-6926-2/20/$31.00 ©2020 IEEE

receptive fields of deeper layers with a novel usage of dilated convolutions. This inspires us that sub-sequences are like 1-D receptive field of TCN, so we borrow the conception of stride to describe the interval of sub-sequences.

Meanwhile, since there are multiple layers in SHRNN, the information flow between layers can have different variants. We know that in CNN the output of one layer is usually average-pooled or max-pooled before being sent to next layer. Similarly, for the hidden states of SHRNN generated by recursive blocks can be refined(e.g. by self-attention [2]) before being sent to next layer. In SHRNN, we call the process Inter-Layer Connection.

Usually, the output of the top layer is used as the final sequence representation. Inspired by fasttext [24], we think simply averaging over all hidden states of one middle-layer may generate good representation. So we propose a layer-wise attention mechanism (LAM) to integrate intermediate representation of middle-layers. The intermediate representation can also be used for classification individually, serving as a regularization term for the loss function. We call this mechanism multi-layer regularization (MLR).

We will describe the design of components involved above in detail, and explore the best practice of SHRNN by large-scale experiments on multiple datasets of text and image classification. Our contributions are mainly composed of two parts:

- We propose a novel model called SHRNN which can outperform the start-of-the-art models in both text and image sequence classification tasks on 8 datasets.
- In order to have a better understanding of SHRNN, we perform extensive experiments to investigate different components of SHRNN and discuss them deeply. These researches provide best practice to construct SHRNN.

## II. Sliding Hierarchical Recurrent Neural Network (**SHRNN**)

### A. Model Overview

Figure 1 shows the overview of the SHRNN architecture. SHRNN mainly consist of 4 parts: 1) Recursive Block, which is the fundamental part of SHRNN, it can slice the input sequence into many sub-sequences, and use RNN (we use LSTM in this paper) to process sub-sequences; 2) Inter-layer Connection, which receives the output from the recursive block of lower layer and transforms them into high-level features by some ways as input of higher layers; 3) Layer-wise Attention, which is used to enhance the interaction between layers; 4) Multi-layer Regularization, which is served as a regularization term for loss function to further improve model's performance. In the following, we will introduce these four components in details.

### B. Recursive Block

Recursive Block is the key component of SHRNN, with the help of recursive block, our model can work like CNN in parallel. In SHRNN, the input sequence will be sliced into many sub-sequences, each of which is processed by a recursive block. The size of a recursive block is defined as the length of sub-sequence it processes. As shown in Fig 2, for a given sequence $S \in R^{l \times h}$, we set the length of sub-sequence as $b$, the stride as $s$, then the overlap between sub-sequence is $b-s$. Tokens which overlap are the repetitive information between the current sub-sequence and previous one, and they are useful to enhance the context information. Intuitively, processing of recursive block is similar to sliding a window in a CNN with some stride.

If a SHRNN has $k$ layers, for the $ith$ layer ($i \in [1, k]$) with sub-sequence size is $b_i$, stride is $s_i$, the input sequence from previous layer is $M$, the length of $M$ is $l_{i-1}$, recursive block can segment the input sequence into $l_i$ sub-sequences, the length of each sub-sequence is $b_i$, and the $l_i$ can be calculated as follows:

$$l_i = Max([\frac{l_{i-1} - b_i - 1}{s_i}] + 1, 0) + 1, \tag{1}$$

where $Max$ is the maximum number between two given numbers, [] is rounding down. Then for the $jth$ sub-sequence in $ith$ layer $sub_i^j \in R^{b_i \times h_{i-1}}$, a LSTM layer with $h_i$ hidden units will encode the $sub_i^j$, getting the encoding representation $\hat{sub}_i^j$, where $\hat{sub}_i^j \in R^{b_i \times h_i}$:

$$\hat{sub}_i^j = LSTM(sub_i^j). \tag{2}$$

### C. Inter-layer Connection

Inter-layer connection determines what kind of information can be passed to the next layer. We propose four different ways for inter-layer connection: Average Operation (also can be seen as 1D-AveragePooling), Max Operation (also can be seen as 1D-MaxPooling), Operation (which is to use the output from $\hat{sub}_i^j$ with attention mechanism as the representation), and Last Operation (which is to use the last vector of $\hat{sub}_i^j$ as the representation).

After inter-layer connection, the $jth$ recursive block in $ith$ layer will output a high-level feature $f_i^j$, and finally generate a feature matrix $F_i$, where $F_i \in R^{l_i \times h}$, as shown in the following equation, where ; is vector concatenation by row:

$$F_i = [f_i^1; f_i^2; ..., f_i^{l_i}], \tag{3}$$

The feature matrix $F_i$ will be passed to the next layer.

### D. Layer-wise Attention

The layer-wise attention is designed to reuse each layer's output to enhance the interaction within layers. The feature matrix of each layer $F_i$ not only passes to the next layer, but an average pooling is also used to compress its information to calculate attention weights:

$$\hat{F}_i = AveragePooling(F_i), i \in [1, k], \tag{4}$$

where $\hat{F}_i \in R^{1 \times h}$, which is the compressed feature obtained from $ith$ layer. Then SHRNN concatenates all the compressed features into a compressed feature matrix $C$, where

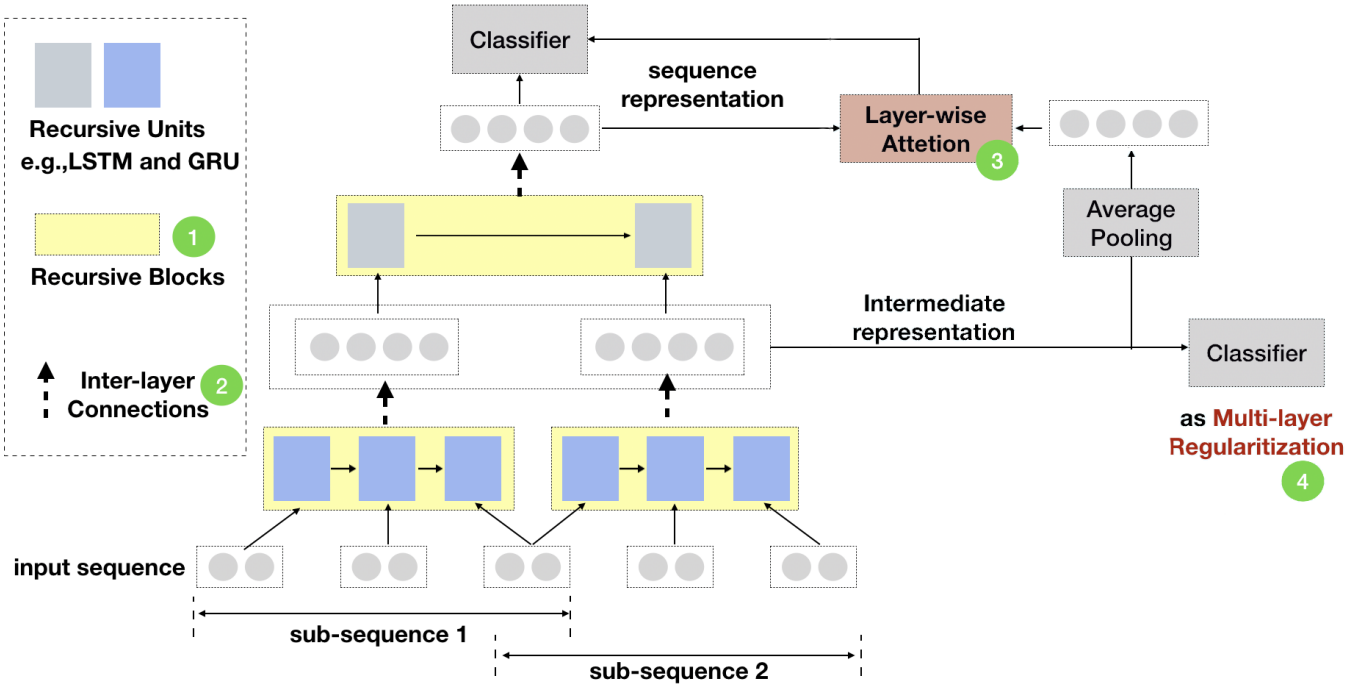


Fig. 1. The overview of 2 layer SHRNN with recursive block size 3, stride 2.

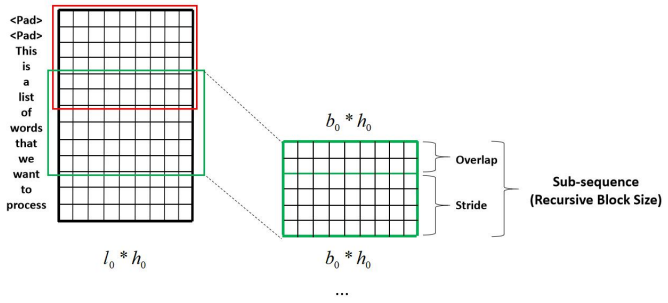


Fig. 2. This picture illustrates the slicing operation, and the visualization of recursive block size, stride and overlap.

$C \in R^{k \times h}$. Next, the soft attention mechanism is used to extract the important information on $C$. SHRNN computes the unnormalized attention weights with the following equation.

$$e_i = Tanh(\hat{F}_i \odot u^T), i \in [1, k], \tag{5}$$

$u$ is the weight vector that we use to compute the unnormalized attention weights. It is randomly initialized and will be adjusted during training. $e_i$ is the unnormalized attention weights, $\odot$ is the dot product between the two given vectors. Besides, in this equation, we use $Tanh$ as the activation function. Next, we use $softmax$ to get the normalized attention weights.

$$\alpha_i = \frac{Exp(e_i)}{\sum_{i=1}^{k} Exp(e_i)}, i \in [1, k], \tag{6}$$

where $\alpha_i$ is the normalized attention weights, and then concatenate $\alpha_i$ as normalized weight vector $W = (e_1, e_2, ..., e_k)$, where $W \in R^{1 \times k}$. Finally we use matrix multiplication to get the output from the attention mechanism.

$$O_{att} = W \times C, \tag{7}$$

where $W \in R^{1 \times k}$ and $C \in R^{k \times h}$, so $O_{att} \in R^{1 \times h}$. SHRNNs use $O_{att}$ to predict the final results.

$$prediction = softmax(O_{att}). \tag{8}$$

### E. Multi-layer Regularization

We use multi-layer regularization as a regular term for the loss function to further improve the model's performance. Multi-layer regularization uses $\hat{F}_i$ from each layer to predict the results, which can reduce the overfitting risk and give SHRNN a stronger generalization ability. The total loss function in SHRNN is as follows:

$$Loss = \alpha_0 loss_{att} + \sum_{i=1}^{k} \alpha_i loss_i . i \in [1, k], \tag{9}$$

where $loss_{att}$ is the loss when $O_{att}$ is used to predict the results, and $loss_i$ is the loss when $\hat{F}_i$ is used to predict the results. $\alpha_0, \alpha_1, ..., \alpha_k$ are the weights of the $k+1$ losses. In our experiments, we find that $\alpha_0 = \alpha_1 = ... = \alpha_k$ yields the best results.

### F. Description of SHRNN Variants

For better understanding, We termed SHRNN without LWA and MLR as base-SHRNN. It is worth noticing that HAN [2],

SRNN [21] and BPIE-SRNN [22] are the particular cases of base-SHRNN. If base-SHRNN divides the the long sequence by sentences and paragraphs and set the inter-layer connection as Attention, it can be used as HAN. In this case, the recursive block is dynamically adjusted. In order to express this slice type, block size is set to -1 and stride is set to 0. If base-SHRNN has equal block size and stride, it can be treated as SRNN. Similarly, base-SHRNN can be used as BPIE-SRNN if we set stride smaller than block size.

## III. Datasets and Experimental Settings

### A. Datasets

- **Yelp reviews.** Yelp Reviews datasets are obtained from the Yelp Dataset Challenge. The datasets Yelp_2013 and Yelp_2014 include 468,608 and 670,440 documents respectively. They all have 5 different labels. We also use the polarity dataset which contains 598,000 documents with binary sentiment labels, named as Yelp_P. Following [21], we used 80% data for training, 10% for validation and 10% for testing. As for Yelp_F, there are 650,000 as training set and 50,000 as test set with 5 different labels.
- **Amazon_F reviews.** Amazon_F is a dataset aiming to predict the evaluation scores of E-commerce reviews, the labels fall into five different classes. Amazon_F has 3,000,000 training documents and 650,000 test documents.
- **IMDB.** IMDB [29] is a movie review dataset for binary sentiment classification, which consists of 50,000 reviews from the Internet Movie Database, with 25,000 training instances and 25,000 test instances.
- **Sequential MNIST and P-MNIST.** MNIST [30] is a classical dataset for image classification, with 60,000 hand-written digits as training set and 10,000 as test set. Each digit is a $28 \times 28$ image. In Sequential MNIST task, MNIST images are presented as a $784 \times 1$ sequence for digit classification, which is frequently used to evaluate a model's ability to retain information from the distant past. P-MNIST is a more challenging setting of Sequential MNIST, in which the sequential order is permuted randomly. P-MNIST also has 60,000 training instances and 10,000 test instances.

### B. Experimental Settings

In order to reduce the randomness of the results, all the experimental results are the average value after five times training. All models are implemented using Keras, and trained on a GPU server with an NVIDIA GTX 1080Ti GPU. The model parameters and the training settings are as follows: (1) for text data sequence classification tasks, we padded the input sentence length as 256. In SHRNN with 2 blocks, the recursive block sizes are [12, 8], and the strides are [3, 2]. In SHRNN with 3 blocks, the recursive block sizes are [12, 8, 8], and the strides are [3, 2, 2]. In SHRNN with 4 blocks, the recursive block sizes are [12, 12, 8, 8], and the strides are [3, 3, 2, 2]; (2) for sequential MNIST and P-MNIST tasks, because the sequence is much longer (768 each) than text data, we implemented SHRNN with 5 blocks, the recursive block sizes are [12, 12, 8, 8, 8], and the strides are [4, 3, 3, 2, 2]; (3) pre-trained Glove embeddings (300-D Glove 840B) [31] are used to initialize the word embeddings; (4) all the recurrent units are set as bidirectional LSTM with 250 hidden units, and we set 2 layers LSTM as a baseline; (5) all models are learned using the RMSProp optimizer with learning rate $\alpha = 0.001$, dropout rate $= 0.2$ for the fully connected-layer, and the batch size is 200 for all.

## IV. Experimental Results and Analysis

In this section, we first have a comprehensive result analysis on different sequence classification tasks. Then according to experimental results, we discuss them deeply by focusing on 5 aspects, that is, the ablation experiments of different components, the relation between recursive block size and stride, the impact of different inter-layer connection types, the analysis of slicing type of recursive block, and the trade-off between efficiency and effectiveness.

### A. Main Results on Different Sequence Classification Tasks

In this subsection, we compare the performance of different SHRNNs and previous models on a range of sequence classification tasks, including text classification (document classification task and sentiment analysis task), and sequential image classification (sequential MNIST and P-MNIST tasks), which are the most comprehensive sequence classification tasks to evaluate the performances of different models.

- **Text Classification Tasks.**
  Table I shows the accuracies between SHRNNs and the state-of-the-art models on 6 different text datasets. From it we can see that 4L-SHRNN outperform all the previous models, including CNN-based model, RNN-based model and attention-based model among all text data sequence classification tasks. The best accuracies finally reach up to 69.6% for Yelp_2013, 72.4% for Yelp_2014, 97.0% for Yelp_P, 67.0% for Yelp_F, 63.8% for Amazon_F and 91.7% for IMDB. It is obvious that SHRNNs can capture a more accurate representation of sequence.
  From Table I we find that SHRNNs can achieve a much better performance than LSTMs, which means SHRNN is a better choice than LSTM for sequence classification tasks. We also find that with the increase of layer number, the SHRNNs can achieve better results in all datasets. That means deeper SHRNN can obtain more useful and abstract representation, leading a better results.
- **Sequential Image Classification Tasks.**
  Table II shows the accuracies of SHRNNs and the state-of-the-art models on sequential MNIST and P-MNIST datasets. For the sequential MNIST task, even if TCN already reaches 99.0% accuracy, SHRNNs can still improve the accuracy by 0.2%. As for the more difficult task P-MNIST, both 4L-SHRNN and 5L-SHRNN can achieve better results than the start of the art, and 5L-SHRNN can obtain the best result with the accuracy is 97.5%. The

TABLE I

ACCURACIES OF SHRNNS AND THE STATE-OF-THE-ART MODELS ON 6 DIFFERENT TEXT DATASETS. (*) MEANS THIS MODEL IS IMPLEMENTED BY THE AUTHORS TO EVALUATE THE PERFORMANCE ON THE GIVEN DATASET. IN THIS TABLE, 2L-SHRNN MEANS SHRNN WITH 2 LAYERS. RESULTS BETTER THAN THE STATE-OF-THE-ART ARE IN BOLD, AND THE VALUE IN BRACKET IS THE IMPROVEMENT COMPARED TO SOTA.

| Model | Yelp_2013 | Yelp_2014 | Yelp_P | Yelp_F | Amazon_F | IMDB |
|---|---|---|---|---|---|---|
| **LSTM** | 66.7 | 70.5 | 94.7 | 65.5 | 61.4 | 89.3 |
| **HAN** [2] | 68.2 | 70.5 | 95.8 (*) | 64.7 (*) | 63.6 | - |
| **LSTM, fixed skip** [25] | - | - | - | - | - | 89.6 |
| **Discriminative-LSTM** [26] | - | - | 95.7 | 63.0 | 61.6 | - |
| **Self-Attention** [27] | - | - | 94.9 | 63.4 | 59.8 | - |
| **Densely Connected CNN+Multi-scale Feature Attention**1001[28] | - | - | 96.5 | 66.0 | 63.0 | - |
| **LSTM with dynamic skip** [14] | - | - | - | - | - | 90.1 |
| **SRNN** [21] | 67.0 | 70.8 | 96.0 | 64.9 (*) | 61.6 | 89.6(*) |
| **BPIE-BiSRNN** [22] | 68.0 | 71.6 | 96.8 | 65.2 (*) | 62.4 (*) | 89.9(*) |
| **2L-SHRNN (ours)** | **68.7(+0.7)** | **71.9(+0.3)** | 96.6 | 65.7 | 63.2 | **90.8(+0.9)** |
| **3L-SHRNN (ours)** | **69.1(+1.1)** | **72.2(+0.6)** | **96.8(+0.0)** | **66.2(+0.2)** | 63.3 | **91.2(+1.3)** |
| **4L-SHRNN (ours)** | **69.6(+1.6)** | **72.4(+0.8)** | **97.0(+0.2)** | **67.0(+1.0)** | **63.8(+0.2)** | **91.7(+1.8)** |

TABLE II

ACCURACIES OF SHRNNS AND THE STATE-OF-THE-ART MODELS ON SEQUENTIAL MNIST AND P-MNIST DATASETS. ACCURACIES HIGHER THAN THE STATE-OF-THE-ART ARE IN BOLD, AND THE VALUE IN BRACKET IS THE IMPROVEMENT COMPARED TO SOTA.

| Model | mnist | p-mnist |
|---|---|---|
| **LSTM** | 98.3 | 91.5 |
| **uRNN [32]** | 95.1 | 91.4 |
| **Zoneout** [33] | - | 95.9 |
| **Dilated GRU** [34] | 99.0 | 96.7 |
| **Skip GRU** [13] | 97.6 | - |
| **r-LSTM Full BP** [15] | 98.4 | 95.2 |
| **TCN** [23] | 99.0 | 97.2 |
| **4L-SHRNN(ours)** | 98.9 | **97.2(+0.0)** |
| **5L-SHRNN(ours)** | **99.2(+0.2)** | **97.5(+0.3)** |

TABLE III

THE ABLATION EXPERIMENTS OF DIFFERENT COMPONENTS IN 4L-SHRNN USING 5 DIFFERENT TEXT CLASSIFICATION DATASETS, LWA MEANS LAYER-WISE ATTENTION, MLR MEANS MULTI-LAYER REGULARIZATION, AND RB MEANS RECURSIVE BLOCK. VALUE IN BRACKET IS THE DECREMENT COMPARED TO SHRNN

| Model | Yelp_2013 | Yelp_2014 | Yelp_P | Amazon_F | IMDB |
|---|---|---|---|---|---|
| SHRNNs | 69.6 | 72.4 | 97.0 | 63.8 | 91.7 |
| - MLR | 69.3(-0.3) | 72.1(-0.3) | 96.6(-0.4) | 63.3(-0.5) | 91.4(-0.3) |
| - LWA | 68.6(-1.0) | 71.8(-0.6) | 96.3(-0.7) | 63.0(-0.8) | 90.9(-0.8) |
| -RB | 66.9(-2.7) | 70.4(-2.0) | 94.5(-2.5) | 61.4(-2.4) | 89.6(-2.1) |

results prove that SHRNN is able to handle sequential image classification tasks really well.

### B. Ablation Experiments

To explore the effectiveness of recursive block, layer-wise attention and multi-layer regularization, we conduct various experiments using 4L-SHRNN with 5 different text classification datasets, the results are shown as bellow.

From Table III we can see that recursive block (RB) plays the most important role in SHRNN, without the help of recursive block, the performances reduce obviously on all datasets, this indicates that recursive block can process the input sequence better than simple LSTM layer. Besides, layer-wise attention (LWA) and multi-layer regularization (MLR) also help SHRNN obtain better results, since LWA can reuse the information from different layers, and MLR can solve the problem of gradient problem in some extend and help optimization of parameters better.

### C. Size of Recursive Block and Stride

In this section, we will analyze the effect of recursive block size and stride. As we can see, the recursive block sizes and strides are independent for each layer, so there are too many possible combinations. In order to have a clear and reasonable comparison, we built various kinds of 2-layer SHRNNs. In details, we conducted a series of experiments whose block sizes are fixed. As shown in Table IV, for a given SHRNN, the parameters are the same within layers, the values of block size are 10, 20, 30, and the values of overlap are 0%, 20%, 40%, 60% and 80% of recursive block size.

- For a given recursive block, we can see the accuracy increases at first, which is due to the benefit of retaining some context information from previous sub-sequence in overlap. However, if the stride is too large, it will bring redundant information, and produce a slightly negative effect on SHRNNs. According to the results, when the values of overlap are 40% or 60% of recursive block size, we can always get the best results with a intermediate value of stride.
- The recursive block size is like the kernel size in CNN, which should have a best practice value. In Table IV we find that for the medium length sequence data (like Yelp_F, which we padded the text length to 256), the recursive block size should not be too large. In contrast, for the long sequence data (like P-MNIST whose length is 768), we set a large recursive block size to obtain more precise long-term dependencies.
- In fact, if the recursive block size is too small, too many sub-sequences will be generated, leading to a risk of memory overflow; and the last layer will receive too many sub-sequences, which results in a poor ability of capturing long-term dependencies after average-pooling. In contrast, if the recursive block size is too large, SHRNN

TABLE IV
THE ACCURACIES USING DIFFERENT SIZES OF RECURSIVE BLOCK AND STRIDE ON DIFFERENT DATASETS. THE RATIO IS THE PROPORTION OF OVERLAP TO RECURSIVE BLOCK SIZE. THE BEST RESULTS ARE IN BOLD.

| Block size | Stride | Overlap (Ratio) | Yelp_F | P-MNIST |
|---|---|---|---|---|
| [10, 10] | [10, 10] | 0 (0%) | 65.2 | 95.9 |
| [10, 10] | [8, 8] | 2 (20%) | 65.4 | 95.8 |
| [10, 10] | [6, 6] | 4 (40%) | 65.2 | **96.1** |
| [10, 10] | [4, 4] | 6 (60%) | **65.7** | 96.0 |
| [10, 10] | [2, 2] | 8 (80%) | 65.4 | 95.5 |
| [20, 20] | [20, 20] | 0 (0%) | 65.2 | 96.0 |
| [20, 20] | [16, 16] | 4 (20%) | 65.5 | 96.1 |
| [20, 20] | [12, 12] | 8 (40%) | **65.8** | **96.6** |
| [20, 20] | [8, 8] | 12 (60%) | 65.7 | 96.5 |
| [20, 20] | [4, 4] | 16 (80%) | 64.6 | 95.6 |
| [30, 30] | [30, 30] | 0 (0%) | 65.1 | 96.5 |
| [30, 30] | [24, 24] | 6 (20%) | 65.4 | 96.8 |
| [30, 30] | [18, 18] | 12 (40%) | **65.6** | 96.5 |
| [30, 30] | [12, 12] | 18 (60%) | **65.6** | **97.1** |
| [30, 30] | [6, 6] | 24 (80%) | 65.5 | 96.0 |

TABLE V
THE ACCURACIES ON VARIOUS DATASETS USING DIFFERENT INTER-LAYER CONNECTION WAYS, OTHER SETTINGS ARE THE SAME. THE BEST RESULTS ARE IN BOLD.

| Inter-layer Connection Ways | Yelp_2013 | IMDB | MNIST | P-MNIST |
|---|---|---|---|---|
| **Average** | 69.1 | 90.9 | 98.9 | 96.4 |
| **Max** | 68.7 | 91.4 | 98.8 | 93.9 |
| **Attention** | 69.3 | 91.0 | 99.0 | 97.0 |
| **Last** | **69.6** | **91.7** | **99.2** | **97.5** |

is very shallow (generally two layers), and performance is also affected.

### D. Different Types of Inter-layer Connection

In this subsection, we will discuss the impact of different connection types between layers. We have an evaluation of 4 different inter-layer connection ways, which are Average Operation, Max Operation, Attention Operation, and Last Operation.

We evaluated these four different ways on 4 different sequence classification tasks. All the models in this subsection have the same parameters except the inter-layer connections. From Table V we can see that the Last Operation always achieves the best results among them, which indicates that the Last Operation is the best and reliable way. As for the Attention Operation, it has competitive results compared with the Last Operation on some tasks. We think the reason why Last is better than Attention Operation in inter-layer connection is that the sub-sequences in SHRNN are usually short, and the Last Operation can obtain enough information without the problem of long-term dependencies, but Attention Operation needs to compute the weights attribution, which may be not reliable when dealing the short sub-sequences sliced from the input sequence. Moreover, we applied LWA to get the high level information between layers, which can be treated as using attention mechanism in a higher level. And for the results of the Max Operation and Average Operation, although they have satisfactory results in document classification and sentiment analysis task, but resulting a poor performance on MNIST and P-MNIST, which shows these ways are unstable.

### E. Slicing Type of Recursive Blocks

In terms of text processing, in order to analyze the difference of slicing type between sentence-length based and fix-length based, we selected various 2L-SHRNNs, and compare it with HAN, on two text datasets Yelp_P and Yelp_F, as shown in Table VI. For a fair comparison, we set recursive block size of SHRNNs to be the average length of sentences, stride to be the same as the recursive block size, so that there is no overlap between recursive blocks, which is the same as HAN. The experimental results show both recursive blocks sliced by sentence and by fixed length achieve similar performance, which is 95.7% vs 95.8% for Yelp_p, and 64.6% vs 64.7% for Yelp_F. It can be seen that although the fixed partition will form an incomplete sequence segment, it does not affect the semantics of itself. The flexible sliding settings allow us to further compensate for the impact of sentence breaks. For example, if we reduce stride by half, the performance of this 2L-SHRNN might increase to 96.0% and 65.0%

### F. The Trade-off between Efficiency and Effectiveness

At last, we will discuss the trade-off between efficiency and effectiveness for different LSTMs and SHRNNs architectures using P-MNIST dataset. The setting of recursive block size and stride can affect the efficiency of the SHRNN directly. Since the block size is usually not too large or too small, basically we adjust its efficiency by changing the number of layers, which is similar to CNN. So we did experiments with different number of layer. The parameters of LSTMs and SHRNNs in this subsection are the same as mentioned above. We report on five indexes: (1) accuracy, (2) time consumption per epoch, (3) epoch number of convergence for each model, and (4) acceleration ratio between each SHRNNs and LSTM (2 layer). From Table VII we can find that:

- It is obvious that SHRNNs have much better performance than LSTMs among all architectures. And as the number of layers increases, both LSTMs and SHRNNs need more time per epoch, but the time consumption of LSTMs has increased significantly, while SHRNNs only have a slight increment. This indicates that SHRNN is applicable to build a deeper architectures for much better results with less time consumption than LSTM;
- As for the epoch number of convergence, we can see that SHRNNs need much less epochs than LSTMs to obtain best results, so the total training time of SHRNNs are much smaller than LSTMs, which is very important in practice;
- For acceleration ratio, the comparison is between each SHRNN and LSTM (2 layer). As we can see in Table VII, we can obtain nearly 3x acceleration ratio compare 2L-SHRNN with LSTM (2 layer), and SHRNN can get a much better result. Even comparing 5L-SHRNN with LSTM (2 layer), we can also have a more than 2x acceleration ratio. From the above all we can see that

TABLE VI

THE ACCURACIES USING DIFFERENT SLICING TYPES IN RECURSIVE BLOCK ON DIFFERENT DATASETS. THE -1 IN HAN MEANS SLICING THE SEQUENCE BY SENTENCES AND PARAGRAPHS. THE AVG-LENGTH MEANS THE BLOCK SIZE IS THE AVERAGE LENGTH OF SENTENCES IN THE GIVEN DATASET.

| Model | Block Size | Stride | Inter-layer Connection | Yelp_P | Yelp_F |
|---|---|---|---|---|---|
| HAN | -1 | 0 | Attention | 95.8 | 64.7 |
| 2L-SHRNN | Avg-length | Avg-length | Attention | 95.7 | 64.6 |
| 2L-SHRNN | Avg-length | 0.5*Avg-length | Attention | 96.0 | 65.0 |
| 2L-SHRNN | Avg-length | Avg-length | Last | 95.9 | 64.5 |
| 2L-SHRNN | Avg-length | 0.5*Avg-length | Last | 96.1 | 65.2 |

TABLE VII

THE TRADE-OFF BETWEEN EFFICIENCY AND EFFECTIVENESS BASED ON DIFFERENT LSTMS AND SHRNNS USING P-MNIST DATASET. **TIME** IS THE TIME CONSUMPTION PER EPOCH, **# EPOCH** IS THE EPOCH NUMBER OF CONVERGENCE FOR EACH MODEL, **ACCEL. RATIO** IS THE ACCELERATION RATIO BETWEEN EACH SHRNN AND LSTM (2 LAYER).

| Model | Accracy | Time | # Epoch | Accel. Ratio |
|---|---|---|---|---|
| **2L-LSTM** | 91.53 | 242s | 10.67 | - |
| **3L-LSTM** | 91.91 | 368s | 11.00 | - |
| **4L-LSTM** | 92.15 | 454s | 12.67 | - |
| **5L-LSTM** | 92.00 | 570s | 12.33 | - |
| **2L-SHRNN** | 96.78 | **84s** | 6.67 | **2.88x** |
| **3L-SHRNN** | 97.03 | 92s | 6.00 | 2.63x |
| **4L-SHRNN** | 97.22 | 100s | 7.00 | 2.42x |
| **5L-SHRNN** | **97.54** | 110s | **5.33** | 2.20x |

SHRNN can obtain much better performances and less time used compared LSTM;

Generally speaking, SHRNN has a satisfactory ability to achieve the efficiency and effectiveness at the same time. Compared with LSTM, SHRNN brings a huge improvement of performance with less time consumption.

## V. CONCLUSION AND FUTURE WORK

We have demonstrated the novel architecture of SHRNN in details, which consists of four collaborative components of Recursive Block, Inter-Layer Connection, Layer-wise Attention and Multi-Layer Regularization. Based on these four components, SHRNN can easily gain improvement over the state-of-the-art models in all 8 datasets for sequence classification tasks. We also showed how to leverage these components by extensive and systematic experiments. Best practices of SHRNN usage can be found in the analysis of each experiment. These best practices will be integrated into a open-sourced and well-encapsulated development library to facilitate the research and usage of SHRNN. Meanwhile, going deeper with RNN, SHRNN provide a novel architecture to handle deeper layers, which termed as main work in our future research.

## VI. ACKNOWLEDGEMENTS

This research was supported by the National Key Research And Development Program of China (No. 2019YFB1405802).

## REFERENCES

[1] O. Yakhnenko, A. Silvescu, and V. G. Honavar, "Discriminatively trained markov model for sequence classification," in *Proceedings of the 5th IEEE International Conference on Data Mining (ICDM 2005), 27-30 November 2005, Houston, Texas, USA*, 2005, pp. 498–505. [Online]. Available: https://doi.org/10.1109/ICDM.2005.52

[2] Z. Yang, D. Yang, C. Dyer, X. He, A. Smola, and E. Hovy, "Hierarchical attention networks for document classification," in *Proceedings of the 2016 conference of the North American chapter of the association for computational linguistics: human language technologies*, 2016, pp. 1480–1489.

[3] D. Tang, B. Qin, and T. Liu, "Document modeling with gated recurrent neural network for sentiment classification," in *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015*, 2015, pp. 1422–1432. [Online]. Available: https://www.aclweb.org/anthology/D15-1167/

[4] R. Ghaeini, S. A. Hasan, V. Datla, J. Liu, K. Lee, A. Qadir, Y. Ling, A. Prakash, X. Z. Fern, and O. Farri, "Dr-bilstm: Dependent reading bidirectional lstm for natural language inference," *arXiv preprint arXiv:1802.05577*, 2018.

[5] B. MacCartney and C. D. Manning, "Modeling semantic containment and exclusion in natural language inference," in *Proceedings of the 22nd International Conference on Computational Linguistics-Volume 1*. Association for Computational Linguistics, 2008, pp. 521–528.

[6] Y. Bengio, P. Simard, P. Frasconi *et al.*, "Learning long-term dependencies with gradient descent is difficult," *IEEE transactions on neural networks*, vol. 5, no. 2, pp. 157–166, 1994.

[7] G.-B. Zhou, J. Wu, C.-L. Zhang, and Z.-H. Zhou, "Minimal gated unit for recurrent neural networks," *International Journal of Automation and Computing*, vol. 13, no. 3, pp. 226–234, 2016.

[8] T. Lei, Y. Zhang, and Y. Artzi, "Training rnns as fast as cnns," *arXiv preprint arXiv:1709.02755*, 2017.

[9] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[10] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder-decoder for statistical machine translation," *arXiv preprint arXiv:1406.1078*, 2014.

[11] M. Miwa and M. Bansal, "End-to-end relation extraction using lstms on sequences and tree structures," *arXiv preprint arXiv:1601.00770*, 2016.

[12] Y. Shen, S. Tan, A. Sordoni, and A. Courville, "Ordered neurons: Integrating tree structures into recurrent neural networks," *arXiv preprint arXiv:1810.09536*, 2018.

[13] V. Campos, B. Jou, X. Giró-i Nieto, J. Torres, and S.-F. Chang, "Skip rnn: Learning to skip state updates in recurrent neural networks," *arXiv preprint arXiv:1708.06834*, 2017.

[14] T. Gui, Q. Zhang, L. Zhao, Y. Lin, M. Peng, J. Gong, and X. Huang, "Long short-term memory with dynamic skip connections," 2019.

[15] T. H. Trinh, A. M. Dai, M.-T. Luong, and Q. V. Le, "Learning longer-term dependencies in rnns with auxiliary losses," *arXiv preprint arXiv:1803.00144*, 2018.

[16] T. Zhang, M. Huang, and L. Zhao, "Learning structured representation for text classification via reinforcement learning," in *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[17] S. E. Hihi and Y. Bengio, "Hierarchical recurrent neural networks for long-term dependencies," in *Advances in Neural Information Processing Systems 8, NIPS, Denver, CO, USA, November 27-30, 1995*, 1995, pp. 493–499. [Online]. Available: http://papers.nips.cc/paper/1102-hierarchical-recurrent-neural-networks-for-long-term-dependencies

[18] W. Chen, Y. Liao, and S. Chen, "Speech recognition with hierarchical recurrent neural networks," *Pattern Recognit.*, vol. 28, no. 6, pp. 795–805, 1995. [Online]. Available: https://doi.org/10.1016/0031-3203(94)00145-C

[19] S. Fernández, A. Graves, and J. Schmidhuber, "Sequence labelling in structured domains with hierarchical recurrent neural networks," in *IJCAI 2007, Proceedings of the 20th International*

*Joint Conference on Artificial Intelligence, Hyderabad, India, January 6-12, 2007*, 2007, pp. 774–779. [Online]. Available: http://ijcai.org/Proceedings/07/Papers/124.pdf

[20] J. Li, M.-T. Luong, and D. Jurafsky, "A hierarchical neural autoencoder for paragraphs and documents," *arXiv preprint arXiv:1506.01057*, 2015.

[21] Z. Yu and G. Liu, "Sliced recurrent neural networks," in *Proceedings of the 27th International Conference on Computational Linguistics*, 2018, pp. 2953–2964.

[22] B. Li, Z. Cheng, Z. Xu, W. Ye, T. Lukasiewicz, and S. Zhang, "Long text analysis using sliced recurrent neural networks with breaking point information enrichment," in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019, pp. 7550–7554.

[23] S. Bai, J. Z. Kolter, and V. Koltun, "An empirical evaluation of generic convolutional and recurrent networks for sequence modeling," *arXiv preprint arXiv:1803.01271*, 2018.

[24] A. Joulin, E. Grave, P. Bojanowski, and T. Mikolov, "Bag of tricks for efficient text classification," *arXiv preprint arXiv:1607.01759*, 2016.

[25] S. Zhang, Y. Wu, T. Che, Z. Lin, R. Memisevic, R. R. Salakhutdinov, and Y. Bengio, "Architectural complexity measures of recurrent neural networks," in *Advances in neural information processing systems*, 2016, pp. 1822–1830.

[26] D. Yogatama, C. Dyer, W. Ling, and P. Blunsom, "Generative and discriminative text classification with recurrent neural networks," *arXiv preprint arXiv:1703.01898*, 2017.

[27] Z. Lin, M. Feng, C. N. d. Santos, M. Yu, B. Xiang, B. Zhou, and Y. Bengio, "A structured self-attentive sentence embedding," *arXiv preprint arXiv:1703.03130*, 2017.

[28] S. Wang, M. Huang, and Z. Deng, "Densely connected cnn with multi-scale feature attention for text classification." in *IJCAI*, 2018, pp. 4468–4474.

[29] A. L. Maas, R. E. Daly, P. T. Pham, D. Huang, A. Y. Ng, and C. Potts, "Learning word vectors for sentiment analysis," in *Proceedings of the 49th annual meeting of the association for computational linguistics: Human language technologies-volume 1*. Association for Computational Linguistics, 2011, pp. 142–150.

[30] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner *et al.*, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[31] J. Pennington, R. Socher, and C. Manning, "Glove: Global vectors for word representation," in *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, 2014, pp. 1532–1543.

[32] M. Arjovsky, A. Shah, and Y. Bengio, "Unitary evolution recurrent neural networks," in *International Conference on Machine Learning*, 2016, pp. 1120–1128.

[33] D. Krueger, T. Maharaj, J. Kramár, M. Pezeshki, N. Ballas, N. R. Ke, A. Goyal, Y. Bengio, A. Courville, and C. Pal, "Zoneout: Regularizing rnns by randomly preserving hidden activations," *arXiv preprint arXiv:1606.01305*, 2016.

[34] S. Chang, Y. Zhang, W. Han, M. Yu, X. Guo, W. Tan, X. Cui, M. Witbrock, M. A. Hasegawa-Johnson, and T. S. Huang, "Dilated recurrent neural networks," in *Advances in Neural Information Processing Systems*, 2017, pp. 77–87.