

Recurrent Neural Network for Text Classification with Hierarchical Multiscale Dense Connections

Yi Zhao¹, Yanyan Shen^{1*} and Junjie Yao²

¹Department of Computer Science and Engineering, Shanghai Jiao Tong University

²School of Computer Science and Software Engineering, East China Normal University

{zhaoyizhaoyi, shenyy}@sjtu.edu.cn, junjie.yao@sei.ecnu.edu.cn

Abstract

Text classification is a fundamental task in many Natural Language Processing applications. While recurrent neural networks have achieved great success in performing text classification, they fail to capture the hierarchical structure and long-term semantics dependency which are common features of text data. Inspired by the advent of the dense connection pattern in advanced convolutional neural networks, we propose a simple yet effective recurrent architecture, named Hierarchical Multiscale Densely Connected RNNs (HM-DenseRNNs), which: 1) enables direct access to the hidden states of all preceding recurrent units via dense connections, and 2) organizes multiple densely connected recurrent units into a hierarchical multiscale structure, where the layers are updated at different scales. HM-DenseRNNs can effectively capture long-term dependencies among words in long text data, and a dense recurrent block is further introduced to reduce the number of parameters and enhance training efficiency. We evaluate the performance of our proposed architecture on three text datasets and the results verify the advantages of HM-DenseRNN over the baseline methods in terms of the classification accuracy.

1 Introduction

Text classification is one of the fundamental tasks in various Natural Language Processing (NLP) applications such as sentiment analysis, topic labeling, and question answering. Recurrent neural networks (RNNs), with the ability of modeling variable length sequential data, have been widely applied to solve the text classification problem [Liu *et al.*, 2016; Yang *et al.*, 2016]. There are two key technical challenges when applying RNNs to classify the semantics of text data. First, the length of texts ranges from a few dozen to several thousand of words. For long text data, the effectiveness of RNNs is known to be comprised due to the problem of exploding and vanishing gradients. Second, text data is typically structured in a hierarchical manner and understanding

its actual semantics needs to fuse the information from different granularities of text components, i.e., words, phrases, sentences. While explicitly modeling hierarchical information of raw texts would have beneficial effects on the classification accuracy, RNNs essentially involve plain structures in sequential order and are thus limited to capture the hierarchical information in text data.

To address the first challenge, various methods have been proposed to capture the long-term dependency among the words in long texts. One line of attempts is the gating mechanism used in LSTM [Hochreiter and Schmidhuber, 1997] and GRU [Chung *et al.*, 2014]. Compared with vanilla RNNs, the gates enable the recurrent architectures to maintain relatively longer memory and thus facilitate the learning of long-term dependencies. Another line of attempts tries to modify the topology of connections among different steps [Zilly *et al.*, 2016; Campos *et al.*, 2017; Chang *et al.*, 2017]. The key idea is to add *skip connections* from early steps to later ones, in order to allow better information and gradient flow by surpassing the middle steps. In practice, the exploding gradient problem can be greatly overcome using the gradient norm clipping strategy [Pascanu *et al.*, 2013], but the vanishing gradient problem still remains to be resolved.

As for the second challenge, explicit boundaries in the parser tree of the texts are exploited using recursive neural structures towards accurate text classification [Tai *et al.*, 2015]. However, the error of text parsing could be propagated to the classification task afterwards. HMRNNs [Chung *et al.*, 2016] use additional boundary variables to automatically construct the hierarchy of the input text, but the training of boundary variables is nontrivial.

In this paper, we introduce a novel hierarchical multiscale densely connected recurrent neural networks (abbrev. HM-DenseRNNs) for text classification. We observe that the *dense connectivity pattern* in the convolutional architectures such as DenseNets [Huang *et al.*, 2017] preserves discriminate information learned from lower layers through a large number of feature transformations, and allows each layer to directly access the gradients from the loss function computed in the final layer. Inspired by this observation, we propose to create dense connections between recurrent units to learn long-term dependencies of words in the text data. Aiming to improve parameter efficiency and speedup the training speed, in each layer, we split the whole recurrent sequence into mul-

*Corresponding author

multiple disjoint *dense recurrent blocks* of the same length k which is a hyperparameter. For a step t within a dense recurrent block, we supply as input the hidden states from the $t - 1$ preceding positions in the same block to the t -th recurrent unit whose hidden state will be updated and forwarded to all the $k - t$ subsequent positions. The dense connectivity within a block has several advantages: (i) preserves information flow to capture long-term semantics dependencies effectively, (ii) allows us to reduce the size of hidden units without compromising much performance, and (iii) enables parameter sharing among the units at the same position across all the blocks to further control model complexity. Besides dense connections, HM-DenseRNNs also stacks multiple DenseRNNs together to form a hierarchical multiscale architecture explicitly. Instead of passing hidden states from lower layers to higher layers at all steps, we only pass information to the upper layer if the step is at the end of a dense recurrent block in the lower layer. As a result, different layers can be updated at different scales, where upper layers are updated at higher time resolution to benefit the memorization of long-term dependencies based on short-term dependencies learned from lower layers.

We evaluate the performance of our proposed HM-DenseRNNs and the variants HM-DenseGRU, HM-DenseLSTM for two text classification tasks: sentiment classification and topic classification. The experimental results show that HM-DenseRNNs achieve better performance and effectively capture long-term dependencies and hierarchical structures in these tasks than various baseline approaches.

2 Related Work

Various approaches have been proposed to address the exploding and vanishing gradient problem in RNNs for sequence learning. One notable line of works tries to modify the inner architecture of RNN cells, such as LSTM [Hochreiter and Schmidhuber, 1997], GRU [Cho *et al.*, 2014]. LSTM uses input, forget and output gates to keep the memory over a long time period and thus can learn long-term dependencies better than vanilla RNNs. GRU simplifies the gates in LSTM and achieves comparable performance in many tasks. These variants of RNNs benefit information flow for long-term dependencies learning and can effectively mitigate the exploding and vanishing gradient problem.

Many other attempts focused on establishing skip connections between the current step and preceding steps in the long past. In Clockwork RNNs [Koutnik *et al.*, 2014], the hidden layer is partitioned into separate modules, each of which is interconnected and updated at its own temporal granularity. DilatedRNN [Chang *et al.*, 2017] constructs different dilated recurrent skip connections for the layers and can be updated with multiple resolutions. The dilations in DilatedRNN is exponentially increasing with the number of layers, which means higher layers are updated less frequently. Recently, a high order recurrent neural network (HORNN) [Soltani and Jiang, 2016] has been proposed to use more memory units to keep track of preceding states in RNNs. At each step, HORNN generates the feedback signal to the hidden layer

by directly combining multiple preceding hidden states. Our work differs from HORNN in two aspects. First, in HORNN, each step is connected to the same number of preceding hidden states, while we use dense recurrent blocks to allow each step connecting to all the preceding steps in the same block. Second, we organize dense recurrent blocks into a hierarchy to form a multi-scale structure. This benefits the learning of inherent hierarchical information in sequences and accelerates the training process. In spite of the above differences, the advantages of skip connections in recurrent settings inspire us to enhance RNNs with a denser connection pattern.

Our work is also related to various hierarchical RNNs. Hierarchical models are widely used in text classification tasks especially the sentiment analysis [Luo *et al.*, 2018]. Hierarchical RNNs [El Hahi and Bengio, 1996] advocate to stack multiple hidden layers in a decreasing order of update frequency, and achieve both computation and training efficiency. This strategy is also adopted in [Chung *et al.*, 2016; Chang *et al.*, 2017], where the higher layers are updated at a slower pace. In this paper, we enhance our densely connected RNNs with a hierarchical architecture so that the units in different layers are updated at different time scales. Moreover, our proposed model updates the higher layer only when the lower layer is at the end of a dense recurrent block. This effectively promotes long-term information flow, reduces model complexity, and allows the updating frequency to decrease exponentially when we go from the lower layers to higher layers.

3 Methodology

We denote the text input as x_1, x_2, \dots, x_T , where $x_t (t \in \{1, \dots, T\})$ stands for the t -th token in the text sequence and T stands for the length of the text sequence. We will use these symbols in the rest of this paper. In this section, we first review the dense connections in DenseNets. We then introduce our densely connected RNNs with the dense connectivity pattern enforced. Finally, we extend DenseRNNs with a new hierarchical multiscale design, and present the application of our proposal on RNN variants.

3.1 Dense Connections in DenseNets

DenseNets [Huang *et al.*, 2017] introduce dense connections to convolutional neural networks, which allow the l -th layer to absorb the feature-maps of all preceding layers as input. Let $\mathbf{x}_1, \dots, \mathbf{x}_l$ denote the feature maps produced by the previous l layers, respectively. The feature map \mathbf{x}_{l+1} from the $l + 1$ -th layer is defined as follows:

$$\mathbf{x}_{l+1} = \mathbf{H}_{l+1}([\mathbf{x}_1, \mathbf{x}_1, \dots, \mathbf{x}_l]) \quad (1)$$

where $[\mathbf{x}_1, \dots, \mathbf{x}_l]$ refers to the concatenation of feature-maps and \mathbf{H}_{l+1} is the nonlinear transformation function defined in layer $l+1$. To facilitate down-sampling in DenseNets, a deep convolutional network is divided into multiple *dense blocks*, where the layers in the same block are connected and equipped with the same feature-map size. This dense connectivity pattern benefits both information and gradient flow. More surprisingly, DenseNets achieve high performance with improved parameter efficiency, i.e., the dense layers are typically very narrow.

3.2 Densely Connected Recurrent Neural Networks

Inspired by the analogy between deep CNNs and RNNs, we aim to adapt dense connections to the recurrent settings. By enforcing direct connections among different steps, we expect to better learn long-term dependencies and allow the model to back-propagate the gradients more effectively.

A simple solution to enforce dense connections in RNNs is allowing the recurrent unit in each step to receive the information learned from all preceding steps. Similar to DenseNets, for the t -th token, we can concatenate the hidden states from all $t - 1$ preceding steps and supply it to the recurrent unit at the t -th step, as input. While this design improves information flow among recurrent units greatly, an important problem is the quadratically increasing number of parameters involved in the units at later steps. Moreover, as the recurrent units with dense connections receive inputs in different lengths, it is difficult to perform parameter sharing among different units, which is one of the key benefits of recurrent architectures.

To control model complexity and enable parameter sharing, we propose to split the whole recurrent sequence into multiple *dense recurrent blocks*, where each block contains direct connections among all the recurrent units involved. Specifically, the concatenation of hidden states from all preceding units in the same block is fed to each recurrent unit, while different blocks are connected in a conventional recurrent manner. Note that in HORNNs [Soltani and Jiang, 2016], each layer is connected to a fixed number L of preceding layers. This symmetry design allows different recurrent units to become indistinguishable, but the maximum direct semantic dependencies cannot exceed $L + 1$ steps. On the contrary, we introduce dense connections to RNNs and allow later recurrent units to get the collective knowledge from previous steps over a larger number of steps. Since different dense recurrent blocks share the same length, the input scale for the recurrent units at the same step across all blocks is identical. As a result, we can perform parameter sharing in block granularity to further control model complexity.

We now formally present our proposal of densely connected RNNs (abbrev. **DenseRNNs**). Consider a sequence of $[\mathbf{x}_1, \dots, \mathbf{x}_T]$ and the length of each *dense recurrent block* is a fixed number k , which is dubbed as *dense depth* in the rest of this paper. For a step $t \in [1, T]$, assume $t = mk + n$ ($m \geq 0, n \in [1, k]$), meaning this step is the n -th step in the $m + 1$ -th dense recurrent block. When $n = 1$, t is the starting step of the block, and the input to this step is $[\mathbf{x}_t, \mathbf{h}_{t-1}]$ to incorporate both sequence data and the last hidden state from the previous (i.e., the m -th) dense recurrent block. When $n > 1$, we concatenate \mathbf{x}_t with the hidden states of all preceding steps in the block, denoted by $\tilde{\mathbf{x}}_t$, as the input to the step. The state of the recurrent unit at step t is updated using the following equations:

$$\tilde{\mathbf{x}}_t = \begin{cases} \mathbf{h}_{t-1}, & \text{if } t \equiv 1 \pmod{k} \\ [\mathbf{h}_{mk+1}, \dots, \mathbf{h}_{mk+n-1}], & \text{otherwise} \end{cases} \quad (2)$$

$$\mathbf{h}_t^l = f(W_1 \mathbf{x}_t + W_2 \tilde{\mathbf{x}}_t + b) \quad (3)$$

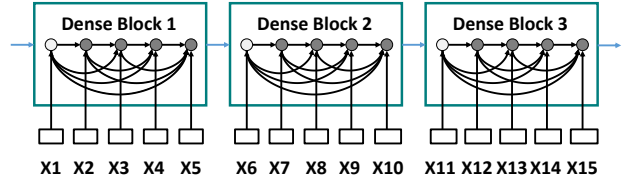


Figure 1: An example of dense recurrent blocks in DenseRNNs. The dense connections are built within each dense block and no transition layer except for a direct connection is constructed between two continuous blocks.

where $f(\cdot)$ is a nonlinear activation function, W and b are the weight matrix and bias, respectively. \mathbf{x}_t is the input at step t , and $\tilde{\mathbf{x}}_t$ is the concatenation of preceding hidden states before the current step in the same dense block. We illustrate the connectivity pattern in DenseRNNs in Figure 1. Note that in our design, we do not include the transition layer (i.e., convolution and pooling) as used in DenseNets. Instead, we directly pass the last hidden state of a dense recurrent block to the next one.

In the original DenseNets, the convolutional kernels used in the same layer from different dense blocks are not shared, in order to learn different kinds of features. In DenseRNNs, we make a trade-off between independent weight matrices at different steps and model complexity. It is not uncommon that many sequences involve more than a few thousand steps. If the parameters for different steps are completely independent, the total number of parameters can easily explode. Therefore, we enable parameter sharing among the units at the same step across all blocks. This is feasible because the steps in the same position of the blocks absorb the same number of preceding hidden states. And this parameter sharing scheme benefits the learning of feature interactions across blocks, which leads to better performance as verified in the experiments.

3.3 Hierarchical Multiscale DenseRNNs

Many sequential data presents an inherent hierarchical structure [Schmidhuber, 1991; Mozer, 1992; El Hahi and Bengio, 1996; Lin *et al.*, 1996; Koutnik *et al.*, 2014]. A typical way to explicitly learning hierarchical representations of sequences is to stack multiple recurrent layers vertically. However, most existing works [Schmidhuber, 1992; El Hahi and Bengio, 1996; Koutnik *et al.*, 2014] used multiscale RNNs, where higher layers are updated at a lower speed than lower layers. The multiscale structure provides several advantages such as computational efficiency and mitigation of vanishing gradients. It also facilitates a more flexible allocation of resources. For instance, we can assign more hidden units to higher layers in order to learn more complex and long-term dependencies.

To capture inherent hierarchical information in sequences, we aim to incorporate DenseRNNs with the hierarchical multiscale architecture. A simple way is to increase dense depth k (the length of a *dense recurrent block*) over layers. Let $k^{(l)}$

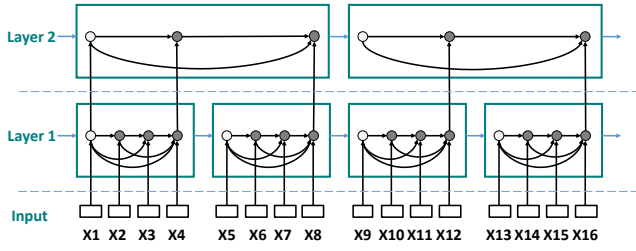


Figure 2: We present a two-layer HM-DenseRNN. The dense depth k in the input layer is 4, and that in the second layer is 8.

denote the dense depth for the l -th layer. We have:

$$k^{(l)} = k_0^{l-1}, \forall l \in [1, L] \quad (4)$$

where k_0 is the dense depth for the first layer. Increasing the size of dense blocks in higher layers allows them to access preceding steps in a wider range. However, because the number of parameters in each layer increases quadratically with dense depth, this method will introduce a huge number of parameters, and will sacrifice training and computation efficiency, unless L is small. We notice that in [Chung *et al.*, 2016], higher layers are not updated at every step, and the updates are determined by the corresponding lower layers. Recall that each dense recurrent block is a basic module to establish dense connections. It is reasonable to expect that the final step in a block obtains a global view of all the preceding steps. Hence, we propose to pass the hidden state of the last step in each block to the upper layer. To be specific, the recurrent units at a higher layer are updated less frequently, only when the corresponding step reaches the end of a dense recurrent block in the lower layer. Formally, consider a step $t = mk^{(l)} + n$, where $m \geq 0, n \in [1, k^{(l)}]$. The update function for the recurrent unit at level l is defined as follows.

$$\tilde{\mathbf{x}}_t = \begin{cases} \mathbf{h}_{t-1}^l, & \text{if } t \equiv 1 \pmod{k^{(l)}} \\ [\mathbf{h}_{mk^{(l)}+1}^l, \dots, \mathbf{h}_{mk^{(l)}+n-1}^l], & \text{otherwise} \end{cases} \quad (5)$$

$$\mathbf{h}_t^l = \begin{cases} f(W_1 \mathbf{h}_t^{l-1} + W_2 \tilde{\mathbf{x}}_t + b), & \text{if } t \equiv 0 \pmod{k^{(l-1)}} \\ \mathbf{h}_{t-1}^l, & \text{otherwise} \end{cases} \quad (6)$$

where $l \geq 1$ and $k^{(l)}$ is length of dense blocks in layer l . \mathbf{h}_t^{l-1} is the input from the below layer, and $\tilde{\mathbf{x}}_t$ is the concatenation of preceding hidden states before the current step within the same block. For the input layer where $l = 0$, we update hidden units at each step as a new sequence input arrives. Note that in the above implementation, the number of parameters will not increase exponentially with the number of layers. This is because we only copy the hidden state of the previous step at most time, which is light-weight.

We dub the extension of DenseRNNs with a hierarchical multiscale structure as **HM-DenseRNNs**. Figure 2 illustrates the architecture of an example HM-DenseRNN.

3.4 Applying Dense Connections in RNN Variants

It is important to note that the dense connectivity pattern in both DenseRNNs and HM-DenseRNNs does not depend

on any specific implementation of recurrent units. Hence, our proposed models are general and can be applied to various RNN variants. For illustration purpose, we describe how to incorporate our proposal into GRU [Chung *et al.*, 2014], which can be easily extended to LSTM [Hochreiter and Schmidhuber, 1997] and other RNN variants.

Consider the dense depth k and a step $t \in [1, T]$. Assume $t = mk + n$, where $m \geq 0, n \in [1, k]$.

DenseGRU

The equations for updating DenseGRU are defined as follows:

$$\tilde{\mathbf{x}}_t = \begin{cases} \mathbf{h}_{t-1}, & \text{if } t \equiv 1 \pmod{k} \\ [\mathbf{h}_{mk+1}, \dots, \mathbf{h}_{mk+n-1}], & \text{otherwise} \end{cases} \quad (7)$$

$$\begin{aligned} \mathbf{z}_t &= \sigma(W_z \mathbf{x}_t + U_z \tilde{\mathbf{x}}_t + b_z) \\ \mathbf{r}_t &= \sigma(W_r \mathbf{x}_t + U_r \tilde{\mathbf{x}}_t + b_r) \\ \tilde{\mathbf{h}}_t &= \tanh(W \mathbf{x}_t + U(\mathbf{r}_t \odot \mathbf{h}_{t-1}) + b) \\ \mathbf{h}_t &= (1 - \mathbf{z}_t) \mathbf{h}_{t-1} + \mathbf{z}_t \tilde{\mathbf{h}}_t \end{aligned} \quad (8)$$

where $\mathbf{z}_t, \mathbf{r}_t, \tilde{\mathbf{h}}_t, \mathbf{h}_t$ are update gate, reset gate, candidate function and hidden state of GRU, respectively. The W and U terms denote the weight matrices, and the b terms are biases. σ is the sigmoid function and \odot is the element-wise product. $\tilde{\mathbf{x}}_t$ is the concatenation of preceding hidden states before the current step in a block. \mathbf{x}_t is the input sequence at step t .

The extensions of DenseGRU and DenseLSTM with hierarchical multi-scale architectures can be simply achieved in the same way as HM-DenseRNNs. We omit the detailed equations due to redundancy.

4 Experiments

In this section, we evaluate the performance of our proposed HM-DenseRNNs and the variants HM-DenseGRU, HM-DenseLSTM for two text classification tasks: sentiment classification and topic classification¹. In addition to the hierarchical multi-scale version, we also evaluate the advantages of DenseRNNs, DenseGRU and DenseLSTM, which are all single-layer RNN models with dense connections enforced.

4.1 Experimental Settings

Datasets

We test different models on three datasets using two tasks: sentiment classification (on IMDB and SST-5) and topic classification (on AG). The details of the datasets are summarized in Table 1.

- **IMDB:** The IMDB dataset [Maas *et al.*, 2011] is a binary sentiment analysis dataset which contains 50,000 movie reviews from IMDb labeled as positive or negative. The number of positive reviews are the same as the negative ones. There are 25,000 movie reviews in the training set and the other 25,000 reviews in the test set. In our experiments, we extract 2,500 examples from the original 25,000 training set for validation.

¹The source codes of our proposed methods are available in <https://github.com/zhaoyizhaoyi/hm-denserrns>

Dataset	# train	# val	# test	# class	# Avg_words
IMDB	22,500	2,500	25,000	2	265
SST-5	8,544	1,101	2,210	5	18
AG	108,000	12,000	7,600	4	45

Table 1: Statistics of datasets.

- **SST-5:** The Stanford Sentiment Treebank [Socher *et al.*, 2013] is a well-established sentiment analysis dataset. We evaluate our models on the fine-grained sentiment classification task which selects one from 5 labels (from very negative to very positive) to classify a movie review. The dataset contains 11,855 sentences, and is split into training (8,544), validation (1,101) and test (2,210) sets.
- **AG:** AG’s news corpus [Zhang *et al.*, 2015] consists of news articles belonging to 4 classes. Each class contains 30,000 examples for training and 1,900 for test. We also randomly select 10% of training examples for validation.

Baselines

We consider both basic RNN models and advanced RNN variants as baselines.

- **Vanilla RNN and Stacked Vanilla RNN.** A single-layer vanilla RNN model and the stack of the 3-layer vanilla RNN.
- **GRU.** Gated Recurrent Unit [Chung *et al.*, 2014] has achieved great success in many sequential tasks. Here we consider both single-layer GRU and 3-layer stacked GRU.
- **LSTM.** Long-short term memory model [Hochreiter and Schmidhuber, 1997] is a widely used recurrent model and has been considered as a baseline to many newly proposed recurrent models. We consider both single-layer LSTM and a 3-layer stacked LSTM.
- **Gated HORNN.** High Order Recurrent Neural Networks [Soltani and Jiang, 2016] has several variants and we select the Gated HORNN which performs best as our baseline. We consider both the single-layer and 3-layer stacked Gated HORNN.

Implementation and Training

We test different number of layers $L = \{1, 3\}$ for HM-DenseRNNs. We also vary the length of dense blocks $k = \{2, 4, 8\}$, according to the specific sequential data in different tasks. In HM-DenseRNNs, dense depth k in the first layer can be larger than 2 (when dense depth is 2, the layer degenerates into a regular RNN). We experiment with different configurations on the number of hidden units in various RNN models. For different tasks, we have tested with $\{32, 64, 128\}$ hidden units. For all the experiments, we deliberately control the number of parameters by adjusting the number of layers and hidden units in order to achieve comparable model complexity.

All the models are implemented with Pytorch. We use Adam [Kingma and Ba, 2014] as the optimizer and decide the initial learning rate from $\{0.01, 0.001, 0.0001\}$ via validation.

The batch size is set adaptively in different tasks. Moreover, we clip the norm of the gradient by a threshold [Pascanu *et al.*, 2013] of 1.0 to prevent gradient exploding. We also perform early stopping if the validation performance is not improved for a number of epochs.

4.2 Main Results

The main results are shown in Table 2. Our proposed models outperform the baselines on all three datasets. HM-DenseGRU achieves the best accuracy on IMDB and SST-5 datasets. HM-DenseLSTM has the highest accuracy on AG’s news dataset. From the results, we have three important observations. First, since the proposed models employ a hierarchical structure which has multi-scale update frequency in each layer, HM-DenseRNNs (i.e., HM-DenseGRU and HM-DenseLSTM) can learn the hierarchical structure of input texts more efficiently and hence outperforms the stacked RNNs (GRU and LSTM included) by a large margin. For instance, on the AG’s news dataset, 3-layer HM-DenseRNN can outperform stacked 3-layer Vanilla RNN by 0.5%. Similarly, 3-layer HM-DenseGRU outperforms the stacked 3-layer GRU by 1.5%, and 3-layer HM-DenseLSTM outperforms the stacked 3-layer LSTM by 1.2%. When taking the parameter size into consideration, we still observe that the 3-layer HM-DenseRNNs outperform the counterparts of stacked 3-layer RNNs with comparable number of parameters. Second, the design of hierarchical multiscale structure is critical to adapt the dense connections into RNN for text classification. As we can see from Table 2, DenseRNNs which simply apply dense connections to single-layer RNNs may have inferior performance compared to vanilla RNNs. Take the IMDB dataset for example. DenseRNN, DenseGRU and DenseLSTM all perform worse than the single-layer RNN, GRU and LSTM, respectively. But HM-DenseRNN (HM-DenseGRU and HM-DenseLSTM) performs much better than the DenseRNN (DenseGRU and DenseLSTM) on all the three datasets, which verifies the effectiveness of the hierarchical multiscale structure. Third, stacked Gated HORNN does not perform unanimously better than the single-layer Gated HORNN, which indicates that it fails to capture the hierarchical structure of the texts. Overall, its performance is better than vanilla RNN and DenseRNN, but worse than GRU (LSTM) and DenseGRU (DenseLSTM). Similar to DenseRNNs, adding connections to previous steps can enhance the performance for simple structure in vanilla RNN, but failing to capture the hierarchical structure of the underlying text limits its performance.

4.3 Ablation Analysis

We also conduct the ablation study to figure out the critical parts of our models. The results on SST-5 dataset are present in Table 3. Similar results are observed on the other two datasets, and we omit them due to redundancy. Note that we mainly consider HM-DenseGRU in this part, but the conclusion also holds on HM-DenseRNN and HM-DenseLSTM. Note that we get DenseGRU by removing the hierarchical part. The results show that all three designs (i.e., hierarchical, multiscale, and dense block) are important. Without any of them, the model performs worse than the single-layer GRU.

Model	# Layers	# Hidden units	# Parameters(k)	IMDB	SST-5	AG
Vanilla RNN	1	64 / 128	23 / 55	81.8 / 83.0	41.4 / 40.5	85.5 / 74.7
GRU	1	64 / 128	70 / 165	90.4 / 90.4	45.8 / 46.4	92.9 / 92.5
LSTM	1	64 / 128	94 / 220	88.8 / 89.3	44.0 / 45.0	92.8 / 93.0
Gated HORNN	1	64 / 128	74 / 181	86.0 / 84.7	42.5 / 42.2	90.6 / 90.6
stacked Vanilla RNN	3	64 / 128	40 / 121	82.1 / 82.5	40.0 / 40.7	89.9 / 90.0
stacked GRU	3	64 / 128	120 / 363	90.3 / 90.2	44.5 / 45.6	91.8 / 90.1
stacked LSTM	3	64 / 128	160 / 484	88.4 / 89.2	43.9 / 41.0	92.3 / 92.3
stacked Gated HORNN	3	64 / 128	132 / 411	87.8 / 83.9	39.7 / 41.1	90.6 / 90.4
DenseRNN	1	64 / 128	48 / 153	76.1 / 82.4	39.9 / 36.3	86.9 / 83.0
DenseGRU	1	64 / 128	144 / 460	89.6 / 89.9	44.9 / 45.1	92.4 / 92.9
DenseLSTM	1	64 / 128	192 / 613	88.0 / 88.2	44.9 / 44.4	92.5 / 93.0
HM-DenseRNN	3	64 / 128	57 / 170	83.9 / 84.3	41.6 / 42.2	90.5 / 90.5
HM-DenseGRU	3	64 / 128	170 / 560	90.6 / 90.4	46.8 / 46.2	92.7 / 93.3
HM-DenseLSTM	3	64 / 128	225 / 746	89.6 / 89.5	45.4 / 45.3	93.2 / 93.5

Table 2: Classification accuracy (%) on the three datasets. The accuracies are the average of 5 random runs.

Model	# Layers	Accuracy(%)
GRU	1	46.4
stacked GRU	3	45.6
HM-DenseGRU	3	46.8
- Hierarchy	1	45.1
- Multiscale	3	46.0
- Dense Block	3	46.3

Table 3: Ablation analysis of HM-DenseGRU on SST-5.

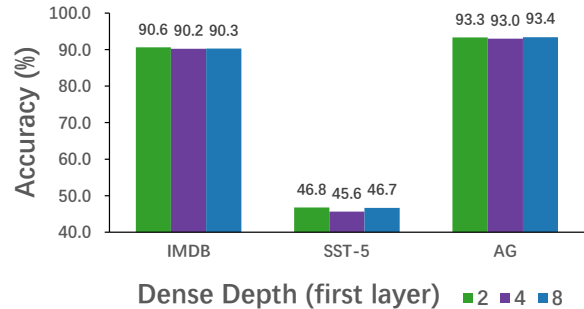


Figure 3: Accuracy with different dense depth of the first layer of HM-DenseGRU.

By combining the three parts into our HM-DenseGRU, we get the best performance.

4.4 Tuning of Hyper-parameters

Dense depth. We assess how the dense depth influences the performance by varying the dense depth of the block in the first layer. According to our design, the dense depth of the block in the upper layer is 2 times the depth of the lower layer. The results are shown in Figure 3. We use HM-DenseGRU as example because the conclusions hold the same for HM-DenseRNN and HM-DenseLSTM. The results indicate that for smaller datasets like IMDB and SST-5 as the dense depth increases, the performance deteriorates, which indicates that starting from normal GRU layer to increase the dense depth is a good choice to avoid potential overfitting problems. But for larger dataset like AG, as the dense depth increases, the performance could become better. However, overall the differences are not significant.

5 Conclusion and Future Work

In this paper, we have proposed a simple yet effective recurrent architecture named DenseRNNs, which borrows the idea of dense connections from DenseNets to the framework of RNNs. We then extend DenseRNNs to a hierarchical version to explicitly model inherent hierarchical structures in text. We introduce a *dense recurrent block* to facilitate dense connections among recurrent units, which allows better information

and gradient flow, benefits the learning of long-term dependencies, and controls model complexity effectively. The hierarchical multiscale DenseRNNs are able to update different layers at different time scales and greatly improve the computation efficiency. The experiments on three text classification tasks have validated the effectiveness of our proposed models, compared with vanilla RNN and its variants.

As future work, we plan to introduce additional decision variables to automatically tune the value of dense depth by the model itself. We also intend to apply the architecture to other RNN variants, and expect further performance gain with a better design of unit structure.

Acknowledgments

This work is supported by the National Key Research and Development Program of China (No. 2018YFC0831604) and NSFC (No. 61602297). Junjie Yao is supported by NSFC 61502169, U1509219 and SHEITC.

References

- [Campos *et al.*, 2017] Víctor Campos, Brendan Jou, Xavier Giró-i Nieto, Jordi Torres, and Shih-Fu Chang. Skip rnn: Learning to skip state updates in recurrent neural networks. *arXiv preprint arXiv:1708.06834*, 2017.
- [Chang *et al.*, 2017] Shiyu Chang, Yang Zhang, Wei Han, Mo Yu, Xiaoxiao Guo, Wei Tan, Xiaodong Cui, Michael Witbrock, Mark A Hasegawa-Johnson, and Thomas S Huang. Dilated recurrent neural networks. In *Advances in Neural Information Processing Systems*, pages 77–87, 2017.
- [Cho *et al.*, 2014] Kyunghyun Cho, Bart Van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259*, 2014.
- [Chung *et al.*, 2014] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.
- [Chung *et al.*, 2016] Junyoung Chung, Sungjin Ahn, and Yoshua Bengio. Hierarchical multiscale recurrent neural networks. *arXiv preprint arXiv:1609.01704*, 2016.
- [El Hahi and Bengio, 1996] Salah El Hahi and Yoshua Bengio. Hierarchical recurrent neural networks for long-term dependencies. In *Advances in neural information processing systems*, pages 493–499, 1996.
- [Hochreiter and Schmidhuber, 1997] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.
- [Huang *et al.*, 2017] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *CVPR*, volume 1, page 3, 2017.
- [Kingma and Ba, 2014] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.
- [Koutnik *et al.*, 2014] Jan Koutnik, Klaus Greff, Faustino Gomez, and Jürgen Schmidhuber. A clockwork rnn. *arXiv preprint arXiv:1402.3511*, 2014.
- [Lin *et al.*, 1996] Tsungnan Lin, Bill G Horne, Peter Tino, and C Lee Giles. Learning long-term dependencies in narx recurrent neural networks. *IEEE Transactions on Neural Networks*, 7(6):1329–1338, 1996.
- [Liu *et al.*, 2016] Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. Recurrent neural network for text classification with multi-task learning. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI 2016, New York, NY, USA, 9-15 July 2016*, pages 2873–2879, 2016.
- [Luo *et al.*, 2018] Ling Luo, Xiang Ao, Feiyang Pan, Jin Wang, Tong Zhao, Ningzi Yu, and Qing He. Beyond polarity: Interpretable financial sentiment analysis with hierarchical query-driven attention. In *IJCAI*, pages 4244–4250, 2018.
- [Maas *et al.*, 2011] Andrew L Maas, Raymond E Daly, Peter T Pham, Dan Huang, Andrew Y Ng, and Christopher 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*, pages 142–150. Association for Computational Linguistics, 2011.
- [Mozer, 1992] Michael C Mozer. Induction of multiscale temporal structure. In *Advances in neural information processing systems*, pages 275–282, 1992.
- [Pascanu *et al.*, 2013] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks. In *International Conference on Machine Learning*, pages 1310–1318, 2013.
- [Schmidhuber, 1991] Jürgen Schmidhuber. Neural sequence chunkers. 1991.
- [Schmidhuber, 1992] Jürgen Schmidhuber. Learning complex, extended sequences using the principle of history compression. *Neural Computation*, 4(2):234–242, 1992.
- [Socher *et al.*, 2013] Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642, 2013.
- [Soltani and Jiang, 2016] Rohollah Soltani and Hui Jiang. Higher order recurrent neural networks. *arXiv preprint arXiv:1605.00064*, 2016.
- [Tai *et al.*, 2015] Kai Sheng Tai, Richard Socher, and Christopher D. Manning. Improved semantic representations from tree-structured long short-term memory networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1556–1566. Association for Computational Linguistics, 2015.
- [Yang *et al.*, 2016] Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard 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*, pages 1480–1489, 2016.
- [Zhang *et al.*, 2015] Xiang Zhang, Junbo Zhao, and Yann LeCun. Character-level convolutional networks for text classification. In *Advances in neural information processing systems*, pages 649–657, 2015.
- [Zilly *et al.*, 2016] Julian Georg Zilly, Rupesh Kumar Srivastava, Jan Koutník, and Jürgen Schmidhuber. Recurrent highway networks. *arXiv preprint arXiv:1607.03474*, 2016.