Densely Connected CNN with Multi-scale Feature Attention for Text Classification

Shiyao Wang[1], **Minlie Huang**[1], **Zhidong Deng**[1*],
[1]State Key Laboratory of Intelligent Technology and Systems
Beijing National Research Center for Information Science and Technology
Department of Computer Science, Tsinghua University, Beijing 100084, China
sy-wang14@mails.tsinghua.edu.cn, aihuang@tsinghua.edu.cn,
michael@tsinghua.edu.cn

Abstract

Text classification is a fundamental problem in natural language processing. As a popular deep learning model, convolutional neural network(CNN) has demonstrated great success in this task. However, most existing CNN models apply convolution filters of fixed window size, thereby unable to learn variable n-gram features flexibly. In this paper, we present a densely connected CNN with multi-scale feature attention for text classification. The dense connections build short-cut paths between upstream and downstream convolutional blocks, which enable the model to compose features of larger scale from those of smaller scale, and thus produce variable n-gram features. Furthermore, a multi-scale feature attention is developed to adaptively select multi-scale features for classification. Extensive experiments demonstrate that our model obtains competitive performance against state-of-the-art baselines on six benchmark datasets. Attention visualization further reveals the model's ability to select proper n-gram features for text classification. Our code is available at: `https://github.com/wangshy31/Densely-Connected-CNN-with-Multiscale-Feature-Attention.git`.

1 Introduction

Text classification is a fundamental problem in natural language processing (NLP). In recent years, deep learning models have been widely applied to this task, including recurrent neural networks (RNN), recursive autoencoders, convolutional neural networks (CNN). In particular, thanks to the great success of CNN in computer vision, CNN has also been successfully applied to NLP applications such as text classification [Kim, 2014; Blunsom *et al.*, 2014; Lei *et al.*, 2015; Zhang *et al.*, 2015; Conneau *et al.*, 2017].

However, most existing CNN models employ convolution filters of fixed size: there is a fixed-size window sliding from the beginning to the end of a text to produce feature maps, which is equivalent to extracting fixed-size n-gram features [Zhang *et al.*, 2015; Conneau *et al.*, 2017; Wang *et al.*, 2017]. This has apparent drawbacks in that text classification needs to extract variable-size features such as phrases to form a better representation [Zhang *et al.*, 2018]. For example, sentiment classification for sentence - "*He's nice to talk to without being patronizing*" (as shown in Figure 1) requires to extract a unigram feature "*nice*" and a trigram feature "*without being patronizing*", which are both positive words or phrases. However, when applying filters of size 1 (see the first row of Figure 1), we obtain one positive feature "*nice*" (green dot) and one negative feature "*patronizing*" (red dot), which is not friendly for classification. When using filters of a larger size (e.g., 3, see the third row of Figure 1), we can obtain an ideal neural response for phrase "*without being patronizing*" (a positive phrase). However, the response for the unigram "*nice*" is undesirably decreased because the model includes additional unnecessary information in either "*He's nice to*" or "*nice to talk*" (see the third row of Figure 1). We term this issue as **the inability to adaptively select multi-scale features in a CNN model for text classification**. Multi-scale features refer to n-gram features with variable n, such as unigram ($n = 1$), bigram ($n = 2$), trigram ($n = 3$) and so on.

A possible solution to this issue is to utilize filters with different window sizes to extract multi-scale features. How-

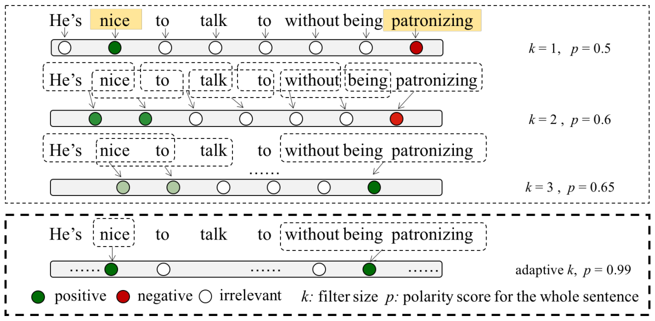

He's nice to talk to without being patronizing
$k = 1$, $p = 0.5$
$k = 2$, $p = 0.6$
$k = 3$, $p = 0.65$
adaptive k, $p = 0.99$
positive
negative
irrelevant
k: filter size p: polarity score for the whole sentence

Figure 1: (Better viewed in color) Comparison between feature representations using convolution filters of fixed window size and of multi-scale window size. The first three rows show feature representations with convolution filters of size 1/2/3, respectively. The last row shows multi-scale feature representations that adaptively select unigram and trigram features.

[*]Corresponding author (Zhidong Deng).

ever, such a solution requires much experimental efforts to find the optimal combinations of different filter sizes. Moreover, this solution is essentially to widen the network with different filter sizes, equivalent to learning several separate, disconnected networks in parallel. Thus, the interactions of feature maps from different filter sizes are not fully exploited. We argue that a deeper model is more elegant than a wider solution in that: a deeper model is more compact because it can construct hierarchical representations by multi-level abstraction and there is no need to re-learn redundant feature maps. As shown in Figure 2(a), the model can construct trigram features equivalently in the third layer (e.g., $f'(x_1, x_2, x_3)$) by stacking another convolutional layer (window size $w = 2$) to the second layer which extracts bigram features (e.g., $f(x_1, x_2)$ and $f(x_2, x_3)$). Thus, feature maps at upstream layers can be effectively re-used at downstream layers. Furthermore, features can be reused more flexibly by adding dense connections between layers (denoted by the red dotted lines in Figure 2). For instance, a trigram feature may be composed not only from $f(x_1, x_2)$ and $f(x_2, x_3)$, but also possibly from x_1 and $f(x_2, x_3)$ (i.e., $x_1(x_2x_3)$). Consequently, such a deep model offers more flexibility to construct multi-scale features than a wide model.

In this paper, we propose a densely connected CNN model with multi-scale feature attention for text classification. To equip the model with the ability to generate multi-scale features, we adopt dense connections between different convolutional layers, inspired by [Huang *et al.*, 2017a]. Through the dense connections, the model can compose features of larger scale from those of smaller scale more flexibly. To select task-friendly features from all the possible multi-scale features, we design an attention module before the classification layer. The two characteristics enable the proposed model to obtain competitive results on several text classification datasets.

In summary, the contributions of this paper include:

- We propose a new CNN model that is equipped with dense connections between convolutional layers and with a multi-scale feature attention mechanism. Through these two design considerations, the model is able to adaptively select multi-scale features for text classification.
- We extensively evaluate the model on six text classification tasks. The model obtains competitive performance compared with state-of-the-art baselines on several benchmark datasets.

2 Related Work

Deep Neural Networks

Recently, deep neural networks have achieved great success in many NLP applications. There are a variety of representation learning models for text representation. Recurrent neural networks (RNN), including long short-term memory (LSTM) and gated recurrent units (GRU) have been widely used in text processing because of their strong performance in processing the structure of text. Several variants are also proposed, such as [Tai *et al.*, 2015; Huang *et al.*, 2017b].

Another type of popular models are undoubtedly CNNs. [Collobert *et al.*, 2011] utilized convolution filters on sliding windows for a text sequence and applied a max operation to capture the most useful local features. [Kim, 2014] adopted multiple filters with different window sizes to extract multi-scale convolutional features for text classification. A dynamic k-max pooling mechanism was proposed in [Blunsom *et al.*, 2014]. [Lei *et al.*, 2015] proposed a novel feature mapping operator to generate non-consecutive n-gram features. [Wang *et al.*, 2017] used a large taxonomy knowledge base to improve performance. Different from taking word vectors as input, [Johnson and Zhang, 2015] applied CNN to learn the embeddings of text regions directly. Based on the text region embeddings, [Johnson and Zhang, 2017] aims to deepen CNNs without increasing much computational cost. All the above approaches are based on word embeddings, while [Zhang *et al.*, 2015] proposed character-level CNN models and reported competitive results. [Xiao and Cho, 2016] utilized both convolutional and recurrent layers to encode character inputs. [Conneau *et al.*, 2017] adopted very deep convolutional neural networks , i.e. ResNet [He *et al.*, 2016] to text classification.

Attention Mechanism

Attention becomes an effective mechanism for selecting significant information to obtain superior results. Deep neural networks, including CNNs and RNNs, can obtain better results if equipped with attention mechanisms. Among many proposed attention mechanisms, there are some noticeable examples including soft and hard attention [Xu *et al.*, 2015], global and local attention [Luong *et al.*, 2015], and source-target attention and self attention [Lin *et al.*, 2017]. Specifically, [Yang *et al.*, 2016] developed a two-level attention mechanism on GRU (i.e. word attention and sentence attention). [Yin *et al.*, 2016] proposed a CNN based attention module for jointly performing attention between two CNN hierarchies. [Lin *et al.*, 2017] developed an attention mechanism for extracting an interpretable sentence embedding.

3 Method

3.1 Overview

The framework of our model is shown in Figure 2. The left panel shows the model in an intuitive view while the right one gives more technical details. The model begins with a text input $x_1x_2 \cdots x_m$, and generates multi-scale features by convolution blocks and through dense connections (denoted by the red dotted lines). The upstream convolutional blocks construct features for small n-grams, and downstream ones for large n-gram features[1]. The dense connections enable the model to compose features of larger scale from those of smaller scale flexibly because downstream blocks have access to all upstream feature maps. To select task-friendly features, an attention mechanism is presented to re-weigh these multi-scale feature maps through the two processes: filter ensemble ($F_{ensem}(\cdot)$) and scale reweight ($F_{reweight}(\cdot)$). A final representation is then built for text classification.

[1]Upstream blocks are close to input while downstream close to output.

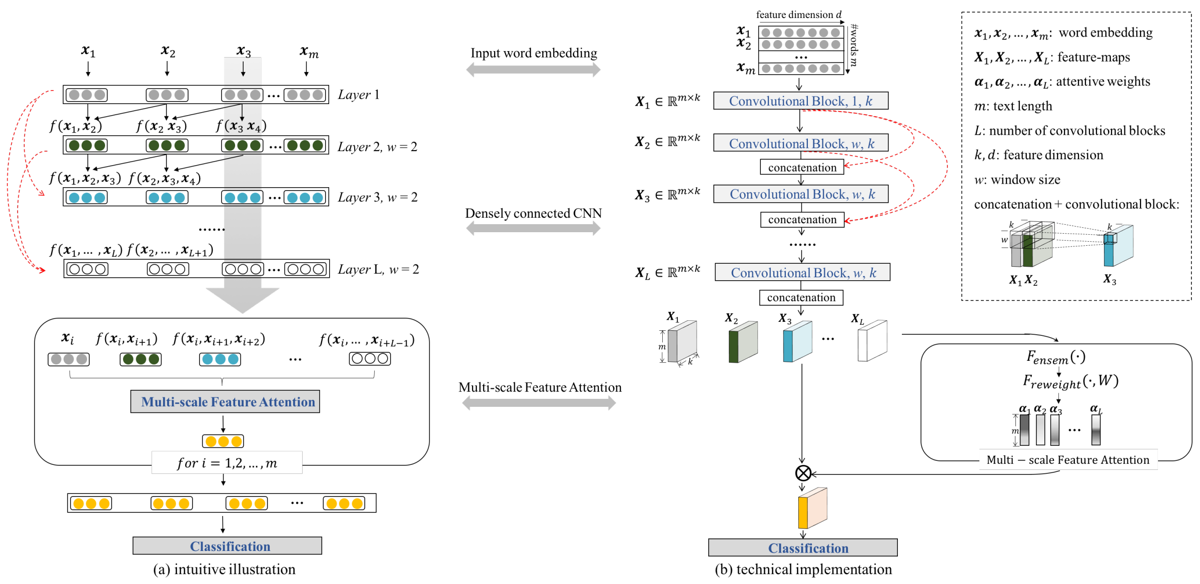


Figure 2: (Better viewed in color) Framework of our densely connected CNN with multi-scale feature attention. (a) Intuitive illustration of how the model generates multi-scale features and how the features are attentively used for classification. (b) Technical implementation, including convolution and concatenation operations, dense connections, and multi-scale feature attention. Red dotted lines indicate dense connections.

3.2 Densely Connected CNN

Preliminary. The proposed model is illustrated in Figure 2. Let $\boldsymbol{x}_i \in \mathbb{R}^d$ be the d-dimensional pretrained word vector of the i-th word in a text and the input text can be represented as a matrix:

$$\boldsymbol{X} = [\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_m]_{m \times d} \tag{1}$$

where m is the number of words in a text. In addition to the input $\boldsymbol{X}$, the outputs of each intermediate layer can be similarly represented as $\boldsymbol{X}_l = [\boldsymbol{x}_l^1, \boldsymbol{x}_l^2, ..., \boldsymbol{x}_l^m]_{m \times k}$, where l ($1 \le l \le L$) is the layer index. L denotes the total number of convolutional blocks and k is the dimension of the transformed feature representation.

A convolutional block generates intermediate feature representations, which can be formulated as below:

$$\boldsymbol{X}_l = f(\mathcal{W}_l, \boldsymbol{X}_{l-1}) \tag{2}$$

where $f(\cdot)$ is a composite function consisting of three cascaded operations: a convolution, a batch normalization (BN) [Ioffe and Szegedy, 2015], and a rectified linear unit (ReLU) [Nair and Hinton, 2010]. $\boldsymbol{X}_l$ is produced by receiving one preceding features $\boldsymbol{X}_{l-1}$ with learnable weights $\mathcal{W}_l \in \mathbb{R}^{k \times w \times k}$. The weight matrix $\mathcal{W}_l$ consists of k filters, each of which is of size $w \times k$, convolving w adjacent vectors (e.g., $\boldsymbol{x}_{l-1}^i$ and $\boldsymbol{x}_{l-1}^{i+1}$ when $w = 2$). This is denoted by "**Convolutional Block, *w*, *k***" in Figure 2. Notice that we use zero padding to the two sides of the input to ensure that the resulting feature map has the same size as the input. For instance, we pad $\mathbf{0} \in \mathbb{R}^k$ after $\boldsymbol{x}_l^m$ when $w = 2$; and pad $\mathbf{0}$ before $\boldsymbol{x}_l^1$ and after $\boldsymbol{x}_l^m$ when $w = 3$.

Traditional CNN models stack convolutional blocks sequentially to form hierarchical representations. For text processing, convolutions can be viewed as to extract n-gram features over a word sequence, and the hierarchical structure enables CNNs to produce features of large scale incrementally. However, only conventional connections may not deal with language composition properly, as illustrated in the introduction. Inspired by [Huang *et al.*, 2017a], we adopt dense connections between convolutional layers to equip the model with the ability to compose representations from multi-scale features (i.e., variable n-grams), which is critical for language processing.

Dense connections. Motivated by the above observation, we adopt a densely connected CNN for text classification, following DenseNet for computer vision [Huang *et al.*, 2017a]. The model takes as input the outputs of all upstream layers $\boldsymbol{X}_1, \boldsymbol{X}_2, \cdots, \boldsymbol{X}_{l-1}$, and generates feature maps for the current layer l, as formulated below:

$$\boldsymbol{X}_l = f(\mathcal{W}_l, [\boldsymbol{X}_1, \boldsymbol{X}_2, ..., \boldsymbol{X}_{l-1}]) \tag{3}$$

where $[\boldsymbol{X}_1, \boldsymbol{X}_2, ..., \boldsymbol{X}_{l-1}]$ refers to the concatenation of the feature-maps produced by layers $1, ..., l-1$ and each $\boldsymbol{X}_i \in \mathbb{R}^{m \times k}$. Weight $\mathcal{W}_l \in \mathbb{R}^{(l-1) \times k \times w \times k}$. As shown in the bottom of the legend of Figure 2, the model produces $\boldsymbol{X}_1$ and $\boldsymbol{X}_2$, then generates features by applying k filters of size $w \times k$ on $\boldsymbol{X}_1$ and $\boldsymbol{X}_2$ respectively, resulting in two $m \times k$ matrices. $\boldsymbol{X}_3$ is obtained by element-wise addition of the two resulting matrices. Notice that we first apply one layer of k filters of size $1 \times d$ to convert the initial feature dimension from d to k. The size of all the feature maps is preserved to the same ($m \times k$) because they will be fed into the attention module which requires the same dimension for multi-scale features.

There is a remarkable advantage in such densely connected CNN: the model computes downstream feature maps (corresponding to larger-scale n-grams) by considering upstream features (smaller-scale n-grams). In this manner, words or shorter phrases will be fully used in obtaining feature maps of longer phrases, resulting in the flexibility of extracting multi-scale features.

3.3 Multi-scale Feature Attention

Through dense connections, the downstream layers of the CNN model has access to the features generated at the upstream layers. However, effective use of these features (some are redundant) is still a key issue. In this subsection, we present a multi-scale feature attention mechanism to effectively use these features for classification. The mechanism adaptively selects features of different scales at each position of a text. As shown in Figure 2(a), at position i, the attention mechanism computes a weight distribution over x_i, $f(x_i, x_{i+1})$ (features from the second layer), and $f(x_i, x_{i+1}, \cdots, x_{i+L-1})$ from the L-th layer. These feature values indicate the responses of multi-scale n-grams, i.e., unigram x_i, bigram $x_i x_{i+1}$,..., and L-gram $x_i x_{i+1} \cdots x_{i+L-1}$.

The attention mechanism contains two operations: **filter ensemble** and **scale reweight**. Filter ensemble aims to develop scalar descriptors s_l^i to represent features of each scale $\boldsymbol{x}_l^i$ at position i. Scale reweight uses the descriptors s_l^i as input and outputs a softmax distribution of attention weights to reweigh these features of different scales, i.e., $\boldsymbol{x}_1^i, \boldsymbol{x}_2^i, \cdots, \boldsymbol{x}_L^i$.

Filter ensemble. As shown in Figure 2, we use k filters in each convolutional block, generating $\boldsymbol{X}_l = [\boldsymbol{x}_l^1, \boldsymbol{x}_l^2, ..., \boldsymbol{x}_l^m]_{m\times k}$. Each $\boldsymbol{x}_l^i \in \mathbb{R}^k$, indexed by i, denotes the k-dimensional features in the i-th position of a text at layer l. We then use a scalar to express each feature vector $\boldsymbol{x}_l^i$ as

$$s_l^i = \mathcal{F}_{ensem}(\boldsymbol{x}_l^i) = \sum_{j=1}^{k} \boldsymbol{x}_l^i(j) \tag{4}$$

where $\mathcal{F}_{ensem}(\cdot)$ indicates a function that sums all k elements of the input vector. The scalar can be used as a descriptor of the feature vector, because $\boldsymbol{x}_l^i$ is yielded by applying k filters on the preceding feature maps and each value in $\boldsymbol{x}_l^i$ is a neural response value. Thus, the sum of all the values in $\boldsymbol{x}_l^i$ can be viewed as feature salience.

Scale reweight. After obtaining s_l^i through filter ensemble, we will use them as input to generate attention weights in order to adaptively re-weight the features from different scales. We define the final representation $\boldsymbol{x}_{atten}^i$ and attention weights α_l^i as follows:

$$\begin{aligned} \boldsymbol{x}_{atten}^i &= \sum_{l=1}^{L} \alpha_l^i \boldsymbol{x}_l^i \\ \sum_{l=1}^{L} \alpha_l^i &= 1, \forall i, 1 \leq i \leq m \end{aligned} \tag{5}$$

where $\boldsymbol{x}_l^i, \boldsymbol{x}_{atten}^i \in \mathbb{R}^k$, and α_l^i are attention weights. Note that feature maps at different layers have correspondence to the scale of features. For instance, when $w = 2$, $\boldsymbol{X}_1$ corresponds to unigram features, and $\boldsymbol{X}_3$ to trigram features.

The attention weights are produced as follows:

$$\begin{aligned} \boldsymbol{\alpha}^i &= softmax(\mathbf{MLP}(\boldsymbol{s}^i)) \\ \boldsymbol{s}^i &= [s_1^i, s_2^i, ..., s_L^i] \\ \boldsymbol{\alpha}^i &= [\alpha_1^i, \alpha_2^i, ..., \alpha_L^i] \end{aligned} \tag{6}$$

where s_l^i is defined by Eq. (4), and $\mathbf{MLP}$ is a multi-layer perceptron.

After being processed by the attention module, the final representation $\boldsymbol{X}_{atten} = [\boldsymbol{x}_{atten}^1, \boldsymbol{x}_{atten}^2, ..., \boldsymbol{x}_{atten}^m] \in \mathbb{R}^{m\times k}$ is generated, which will be fed into the classification layer.

3.4 Objective Function

The training objective is to minimize the cross-entropy loss:

$$\phi = \mathcal{L}(y, h(W, \boldsymbol{X}_{atten})) \tag{7}$$

where $h(W, \boldsymbol{X}_{atten})$ is the predicted output distribution based on the final representation $\boldsymbol{X}_{atten}$ and y is the referenced distribution. $\mathcal{L}(p, q)$ is the cross entropy function between two distributions p and q.

Note that supervision signals are propagated to upstream blocks more straightforwardly through the shortcutd or dense connections. Such connections enforce the upstream layers to learn task-friendly features, also known as "deep supervision" [Huang *et al.*, 2017a].

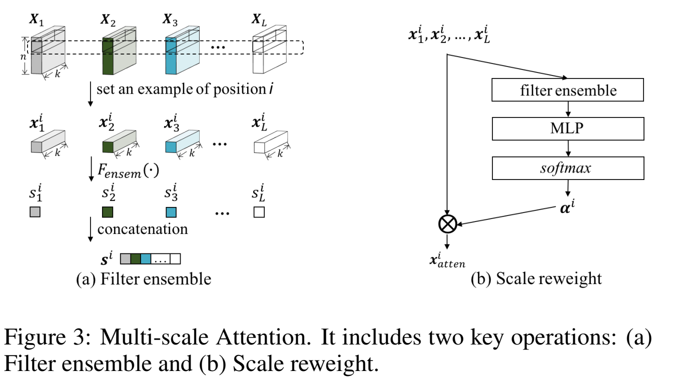


Figure 3: Multi-scale Attention. It includes two key operations: (a) Filter ensemble and (b) Scale reweight.

4 Experiments

4.1 Datasets

We evaluated our model on six public datasets: **MR** from [Pang and Lee, 2005] and the others (**AG, DBPedia, Yelp P./F.**, and **Amazon F.**) from [Zhang *et al.*, 2015]. The statistics of the datasets are summarized in Table 2.

4.2 Implementation details

Input. Word vectors were adopted from [Pennington *et al.*, 2014]. The word embedding was of size 300. The input text was padded to a fixed length - $maxlen$, where $maxlen$ was chosen 50 for MR, 100 for AG and DBPedia, and 300 for Yelp P., Yelp F. and Amazon F., respectively.

Architecture configuration. The model was implemented with Caffe [Jia *et al.*, 2014]. Due to the different lengths of input texts and the number of training samples, we adopted 5 convolutional blocks for MR and AG, and 6 convolutional blocks for the rest datasets. We chose window size $w = 3$

Model		MR	AG	DBP.	Yelp P.	Yelp F.	Amazon F.
Linear	n-gram TF-IDF*	-	92.4	98.7	95.4	54.8	52.4
	FastText* [Grave *et al.*, 2017]	-	92.5	98.6	95.7	63.9	60.2
RNN	LSTM*	77.4	86.1	98.6	94.7	52.5	44.1
	Discriminative-LSTM* [Yogatama *et al.*, 2017]	-	92.1	98.7	92.6	59.6	-
CNN	CNN [Kim, 2014]	81.5	91.6	98.6	93.5	61.0	57.4
	Char-CNN* [Zhang *et al.*, 2015]	-	87.2	98.4	94.7	62.0	59.5
	VeryDeep-CNN* [Conneau *et al.*, 2017]	-	91.3	98.7	95.4	64.7	**63.0**
	Dynamic-Pool* (VeryDeep) [Blunsom *et al.*, 2014]	-	91.3	98.6	95.7	63.0	61.6
	Knowledge-CNN* [Wang *et al.*, 2017]	**83.3**	88.4	-	-	-	-
RNN-CNN	CNN with Recurrent Layer* [Xiao and Cho, 2016]	-	91.4	98.6	94.5	61.8	59.2
Attention	Self-Attention [Lin *et al.*, 2017]	80.1	91.5	98.3	94.9	63.4	59.8
Ours	Densely Connected CNN	80.1	92.9	99.0	96.0	64.5	62.0
	+ Multi-scale Feature Attention	81.5	**93.6**	**99.2**	**96.5**	**66.0**	**63.0**

Table 1: Accuracy of all the models on the six datasets. The results of the baselines marked with * are re-printed from the references.

Dataset	MR	AG	DBP.	Yelp P.	Yelp F.	Amazon F.
Training	9,596*	120K	560K	560K	650K	3M
Testing	1,066*	7.6K	70k	38K	50K	650K
Classes	2	4	14	2	5	5
Avg_words	24	45	55	153	155	93

Table 2: Data statistics. The asterisk (*) means there was no standard training/test split and thus 10-fold cross validation was conducted.

and feature dimension $k = 128$. The classification layer is a two-layer fully connected MLP with ReLU activation function and softmax output. We made slight modifications to the framework shown in Figure 2: pooling layers are added with stride 2 after each convolutional block, which can merge some redundant features and improve computational efficiency. Additional pooling operations with an appropriate stride are required in order to keep the multi-scale features from different layers having the same size.

Training settings. We used stochastic gradient descent (SGD) with a mini-batch of 256. The learning rate is initially set to 0.01 and then gradually decreased to $e - 5$. The training process lasts at most 30 epoches on all the datasets. We applied "L1" regularization and the momentum was set to 0.9.

4.3 Baselines and Main Results

We compared our model with several genres of popular models: linear models [Grave *et al.*, 2017]; RNNs including LSTM and its variants Discriminative-LSTM [Yogatama *et al.*, 2017]; CNNs including classical CNN [Kim, 2014], CNN with dynamic pooling [Blunsom *et al.*, 2014], character-level CNN [Zhang *et al.*, 2015], and very deep CNN [Conneau *et al.*, 2017]; and attention-based models such as [Lin *et al.*, 2017].

Main results. The results are listed in Table 1. The proposed model outperforms the linear models and RNNs on all these datasets. Compared with those shallow but wider CNNs [Kim, 2014], our model has advantages on large datasets (Yelp P./F. and Amazon F.) since the proposed architecture can learn hierarchical and multi-scale features flexibly. Compared to other deep CNNs, the densely connected structure which uses fewer parameters (5-6 convolutional blocks) outperforms the model [Conneau *et al.*, 2017] with 29 convolutional layers. Our model is better than the attention-based model [Lin *et al.*, 2017]. At last, the last two rows show the performance difference with/without multi-scale feature attention, thereby demonstrating the effectiveness of the proposed attention mechanism.

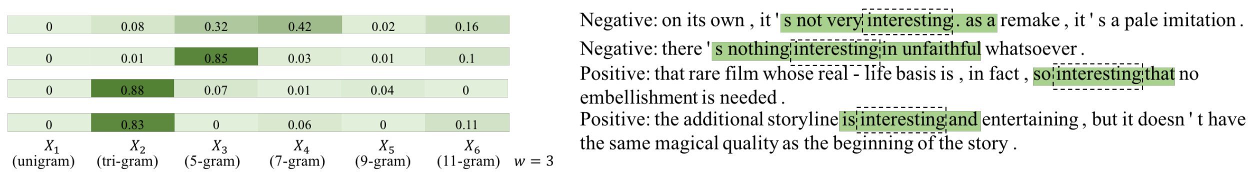
N-gram	Positive Category	Negative Category
trigram	and a wonderful (2) is an enjoyable (3) vividly captures the(15)	is boring . (3) and dull . (9) a waste of (10)
5-gram	it ' s a pleasure (3) the film is an enjoyable (8) the best films of the(10)	of the worst movies of (1) dull , lifeless , and (8) but it doesn ' t (12)
7-gram	of the best films of the year (1) it ' s refreshing to see a (16) can ' t wait to see what (18)	one of the worst movies of the (1) meaningless , vapid and devoid of substance (2) every joke is repeated at least four (3)

Table 3: Examples of top 20 ranking n-grams discovered by our model. The number in brackets denotes the rank.

4.4 Attention Visualization and Phrase Analysis

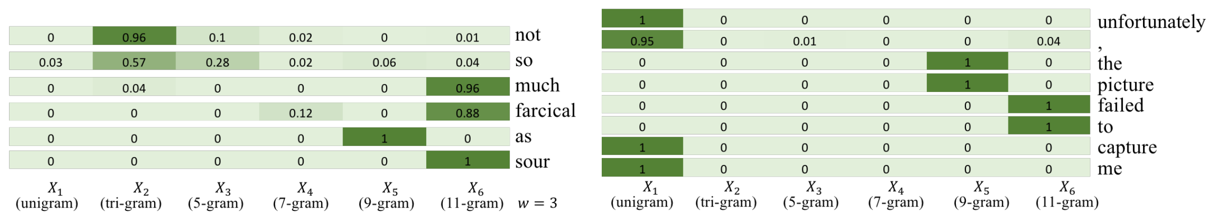
Attention Visualization: Through attention visualization analysis, we made two observations:

1) The same word in different sentences has different optimal feature scales depending on the context. As shown in Figure 4(a), each row indicates a weight distribution over different feature scales $\boldsymbol{x}_l^i (1 \leq l \leq L)$ and i is the position for word "*interesting*". When $w = 3$, $\boldsymbol{X}_l$ corresponds to feature maps of $(2l - 1)$-gram. We presented two negative and two positive sentences. For the negative sentences, the optimal feature scale for the positive word "*interesting*" is larger and includes negators such as "*not*" or "*nothing*", which is more adequate for sentiment classification. Whereas in the other two positive sentences, the optimal feature scale is smaller ($X_2 \sim$ trigram) to form positive phrases. Thus, the model can adaptively select task-friendly feature scales for classification.



Negative: on its own , it ' s not very interesting . as a remake , it ' s a pale imitation .
Negative: there ' s nothing interesting in unfaithful whatsoever .
Positive: that rare film whose real - life basis is , in fact , so interesting that no embellishment is needed .
Positive: the additional storyline is interesting and entertaining , but it doesn ' t have the same magical quality as the beginning of the story .

(a) Word *interesting* forms meaningful features of different scales in different sentences. For the second sentence, the most weighted (0.85) feature is a 5-gram (*s nothing interesting in unfaithful*).



(b) Different words in the same sentence have different optimal scales to form meaningful representation. Self-contained words such as "*unfortunately*" form smaller-scale features, while long phrases form larger-scale features.

Figure 4: Multi-scale attention visualization. The values in a row are a distribution over the feature scales at a word position (see Eq. 5/6).

2) Different words in the same sentence have different optimal scales to form meaningful representations. As shown in Figure 4(a), the optimal feature scale for each word in a sentence tends to be different. For example, in this review "*not so much farcical as sour*", the model retains a smaller scale for "*not*" and chooses larger scales for "*much farcical as sour*". Similar phenomena can be found in the other sentence: a smaller scale for "*unfortunately*" and "," since such words are self-contained, and a larger scale for "*the picture failed to*" to form meaningful representations.

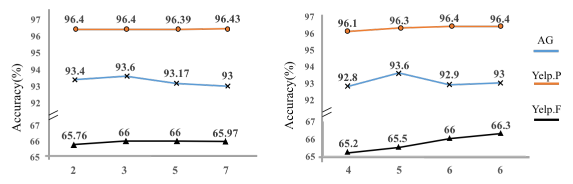
To summarize, the visualization results show that the model is able to adaptively select multi-scale features to form more meaningful representations. Furthermore, these cases show an agreement with our intuition.

Phrase Analysis: Table 3 lists a few phrases discovered by our model on the MR dataset. The phrases are ranked by the product of normalized neural responses and the logarithm of word frequency within each scale. Results show that our model has the ability to select task-relevant phrases, such as "very funny." for positive reviews while "a complete waste of time" for negative reviews. The model may produce bad phrase boundaries, however, this is extremely difficult without structure annotations [Zhang *et al.*, 2018].

4.5 Tuning of Hyperparamters

Window size: We justified how performance is influenced by the window size of convolution filters applied in each block. We varied the window size w among $\{2,3,5,7\}$. As shown in Figure 5(a), the accuracy is not significantly influenced by the window size. This is in line with our claim that the model is able to construct high-level features flexibly from low-level features through dense connections and multi-scale feature attention.

Network Depth: We assessed how the network depth influences the performance by varying the number of convolutional blocks. The results are shown in Figure 5(b). For the



(a) Window Size (b) Network Depth

Figure 5: Accuracy with different window sizes and network depths.

larger datasets (Yelp.F/Amazon.F), as the network depth increases, the performance is further improved since the model is equipped with more learnable weights. For AG, a relatively small dataset, the performance drops slightly when increasing the depth from 5 to 7 blocks, possibly due to over-fitting with too many parameters. In general, increasing network depth is able to obtain better performance on larger datasets, but not that remarkably because of the dense connections we applied. Note that the parameters of the 4th - 7th blocks grow almost linearly, corresponding to 0.74M, 0.97M, 1.25M and 1.60M, respectively.

5 Conclusion

In this paper, we present a new CNN model equipped with dense connections and multi-scale feature attention for text classification. Through dense connections, the model is able to flexibly generate larger n-gram features from variable smaller n-gram features. By multi-scale feature attention, the model can adaptively select task-friendly yet effective features from many multi-scale features for text classification. The model demonstrates competitive performance on six benchmark datasets. Attention visualization reveals that our model can select the optimal scale to form meaningful representation for text classification.

Acknowledgments

This work was supported partly by the National Key Research and Development Program of China under Grant No. 2017YFB1302200, by research fund of Tsinghua University - Tencent Joint Laboratory for Internet Innovation Technology, and by the National Science Foundation of China (NSFC) under Grant No. 91420106, 90820305, and 60775040.

References

[Blunsom *et al.*, 2014] Phil Blunsom, Edward Grefenstette, and Nal Kalchbrenner. A convolutional neural network for modelling sentences. In *ACL*, 2014.

[Collobert *et al.*, 2011] Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12(Aug):2493–2537, 2011.

[Conneau *et al.*, 2017] Alexis Conneau, Holger Schwenk, Loïc Barrault, and Yann Lecun. Very deep convolutional networks for natural language processing. In *EACL*, 2017.

[Grave *et al.*, 2017] Edouard Grave, Tomas Mikolov, Armand Joulin, and Piotr Bojanowski. Bag of tricks for efficient text classification. *EACL (2)*, pages 427–431, 2017.

[He *et al.*, 2016] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, pages 770–778, 2016.

[Huang *et al.*, 2017a] Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *CVPR*, 2017.

[Huang *et al.*, 2017b] Minlie Huang, Qiao Qian, and Xiaoyan Zhu. Encoding syntactic knowledge in neural networks for sentiment classification. *ACM Transactions on Information Systems (TOIS)*, 35(3):26, 2017.

[Ioffe and Szegedy, 2015] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, 2015.

[Jia *et al.*, 2014] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional architecture for fast feature embedding. In *ACM MM*, pages 675–678, 2014.

[Johnson and Zhang, 2015] Rie Johnson and Tong Zhang. Effective use of word order for text categorization with convolutional neural networks. In *NAACL*, pages 103–112, 2015.

[Johnson and Zhang, 2017] Rie Johnson and Tong Zhang. Deep pyramid convolutional neural networks for text categorization. In *ACL*, pages 562–570, 2017.

[Kim, 2014] Yoon Kim. Convolutional neural networks for sentence classification. In *EMNLP*, pages 1746–1751, 2014.

[Lei *et al.*, 2015] Tao Lei, Regina Barzilay, and Tommi S. Jaakkola. Molding cnns for text: non-linear, non-consecutive convolutions. pages 1565–1575, 2015.

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

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

[Nair and Hinton, 2010] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *ICML*, pages 807–814, 2010.

[Pang and Lee, 2005] Bo Pang and Lillian Lee. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *ACL*, pages 115–124, 2005.

[Pennington *et al.*, 2014] Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In *EMNLP*, volume 14, pages 1532–43, 2014.

[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 *ACL*, pages 1556–1566, 2015.

[Wang *et al.*, 2017] Jin Wang, Zhongyuan Wang, Dawei Zhang, and Jun Yan. Combining knowledge with deep convolutional neural networks for short text classification. In *IJCAI*, pages 2915–2921, 2017.

[Xiao and Cho, 2016] Yijun Xiao and Kyunghyun Cho. Efficient character-level document classification by combining convolution and recurrent layers. *arXiv preprint arXiv:1602.00367*, 2016.

[Xu *et al.*, 2015] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. Show, attend and tell: Neural image caption generation with visual attention. In *ICML*, pages 2048–2057, 2015.

[Yang *et al.*, 2016] Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alexander J. Smola, and Eduard H. Hovy. Hierarchical attention networks for document classification. In *NAACL*, pages 1480–1489, 2016.

[Yin *et al.*, 2016] Wenpeng Yin, Hinrich Schütze, Bing Xiang, and Bowen Zhou. Abcnn: Attention-based convolutional neural network for modeling sentence pairs. In *ACL*, pages 259–272, 2016.

[Yogatama *et al.*, 2017] Dani Yogatama, Chris Dyer, Wang Ling, and Phil Blunsom. Generative and discriminative text classification with recurrent neural networks. *arXiv preprint arXiv:1703.01898*, 2017.

[Zhang *et al.*, 2015] Xiang Zhang, Junbo Zhao, and Yann LeCun. Character-level convolutional networks for text classification. In *NIPS*, pages 649–657, 2015.

[Zhang *et al.*, 2018] Tianyang Zhang, Minlie Huang, and Li Zhao. Learning structured representation for text classification via reinforcement learning. In *AAAI*, 2018.