ITERATED DILATED CONVOLUTIONAL NEURAL NETWORKS FOR WORD SEGMENTATION

H. He[*], *X. Yang*[*†], *L. Wu*[‡‡], *G. Wang*[†§]

Abstract: The latest development of neural word segmentation is governed by bi-directional Long Short-Term Memory Networks (Bi-LSTMs) that utilize Recurrent Neural Networks (RNNs) as standard sequence tagging models, resulting in expressive and accurate performance on large-scale dataset. However, RNNs are not adapted to fully exploit the parallelism capability of Graphics Processing Unit (GPU), limiting their computational efficiency in both learning and inferring phases. This paper proposes a novel approach adopting Iterated Dilated Convolutional Neural Networks (ID-CNNs) to supersede Bi-LSTMs for faster computation while retaining accuracy. Our implementation has achieved state-of-the-art result on SIGHAN Bakeoff 2005 datasets. Extensive experiments showed that our approach with ID-CNNs enables 3× training time speedups with no accuracy loss, achieving better accuracy compared to the prevailing Bi-LSTMs. Source code and corpora of this paper have been made publicly available on GitHub[1].

Key words: *deep learning, data mining, intelligent systems applications, parallel and distributed algorithms, pattern recognition*

Received: February 8, 2019 **DOI:** 10.14311/NNW.2020.30.022
Revised and accepted: October 30, 2020

1. Introduction

Many nature languages are not automatically segmented. In other words, there are no space between words, making it hard to process those languages for later artificial intelligence tasks like Information Retrieval and Question Answering etc. Chinese language is such a well known case with completely missing of explicit word delimiters. Therefore, Chinese Word Segmentation (CWS) is a preliminary pre-processing challenge prior to further Chinese language processing tasks.

After [1], CWS has been casted into a sequence tagging problem, involving numerous supervised learning models such as Maximum Entropy (ME) [2] and

[*]Han He; University of Houston Clear Lake, 2700 Bay Area Blvd, Houston TX77059, USA
[†]Xiaokun Yang – Corresponding author; University of Houston Clear Lake, 2700 Bay Area Blvd, Houston TX77059, USA, E-mail: yangxia@uhcl.edu
[‡]Lei Wu; Auburn University at Montgomery, Company Address, Montgomery, AL36117, USA
[§]Guan Wang; Swiss Re Asia Pte. Ltd., Hong Kong Branch Suites 6001-03 & Floor 61, Center Plaza, 18 Harbour Road, Hong Kong, China
[1]https://github.com/hankcs/ID-CNN-CWS

Conditional Random Fields (CRFs) [3, 4]. These early models lack the ability of learning feature representations, instead requiring heavy hand-crafted feature engineering within a fixed size window.

The latest advancement of deep learning has brought fresh air into this challenge. Neural word segmentation approaches arose to reduce efforts in feature engineering. [5] embedded raw character into its vectorial representations as input, adapted the sliding-window based sequence labeling method [6]. [7] extended [5] by exploiting tag embeddings and bigram embeddings. [8] adopted LSTM to capture long-distance preceding context, and an extra fixed-size character window for the short-distance succeeding context. While these models are accurate, the processing speed is limited even using cutting edge modern GPUs, due to the fact that LSTMs are constrained to sequential processing with requiring $O(n)$ time on sentences of length n even under the condition of parallelism.

Instead, Convolutional Neural Networks (CNNs) provide parallelized runtime independent of sequence length [9, 10]. Rather than agglomerating representations token by token, CNNs apply multiple filters in parallel across the entire sequence at once, which only requires $O(1)$ time. Although fast, CNNs suffer from the shortage of effective input width limitation: one token representation is only effective of limited number of input tokens. In single CNN layer, this limitation is the convolution width w. To overcome this issue, one common solution is to stack many CNN layers. At layer l, the size of incorporated context tokens is increased to $r = l(w-1)+1$. This formula states that the number of layers required to capture the whole sequence grows linearly with the length of that sequence. To tackle this scaling, another common approach is the pooling trick. However, though innocuous in sentence classification, the reduction of output resolution using pooling is pernicious for sequence tagging.

In response, [11] applied *dilated convolutions* [20] to sequence labeling. The key of dilated convolutions is to skip over every d inputs, namely dilation width. By stacking layers of dilated convolutions, the effective input width is exponentially increased with dilation width, which is now given by $2^{l+1}-1$.

When l is big, deep neural networks tend to overfit small datasets. [11] proposed *iterated dilated CNNs* architecture (ID-CNNs) to reuse the same block of dilated convolutions. This parameter sharing trick is shown to be effective on a small NER dataset as CoNLL 2003.

Under this context, this paper proposes a novel approach adapting ID-CNNs for CWS. The contributions of this paper could be summarized as:

- Explored a novel neural architecture in replacement of Bi-LSTM.
- Extensive experiments showed the detail advantages over Bi-LSTM in speed and accuracy.
- Extensive studies on state-of-the-art research papers and latest resources of literature reviews [7, 8, 12–16] indicate that our work is the first successful ID-CNNs approach with state-of-art scores and meanwhile achieves superior 3× runtime speed. Finally, we made the source code and corpora publicly available on GitHub[2].

[2] https://github.com/hankcs/ID-CNN-CWS

The organization of this paper is follows: we first review the relevant related papers in Section 2. Our approach is discussed in more detail in Section 3, and in Section 4, the experimental results are discussed. Finally, in the last section we present our conclusion.

2. Related work

In this section, a brief review of Chinese Word Segmentation is discussed.

Chinese Word Segmentation has been in the spotlight of researchers for decades [28]. After the proposal to cast it into a character-based tagging problem by pioneer [1], [4] employed CRFs as a strong sequence labeling model. Later various sequence labeling based works [17, 18, 27, 29] were proposed. Almost whenever a new sequence labeling model is proposed, it will quickly make its way to CWS. These early works require heavy hand-crafted feature engineering within a fixed size window. Their feature engineering part often involves expert knowledge which is domain-dependent and hard to transfer between domains. Feature templates designed by experts usually applies to a fixed size sliding window which hinders the global reasoning ability of their systems. Some of these templates generates overabundance features which results in poor runtime performance.

Recently with new techniques of deep learning, neural word segmentation arose to reduce efforts in feature engineering. Zheng et al. [5] adapted the sliding-window based sequence labeling [6] with character embeddings. Their sliding-window model still cannot capture long distance features. Pei et al. [7] extended Zheng et al. [5]'s work by exploiting bigram features and tag embeddings. Their bigram embeddings suffer from data sparsity issue as the number of bigrams is the squire of the number of characters. Inspired by the success of Recurrent Neural Networks, Chen et al. [8] employed Long Short Term Memory (LSTM) network to capture long-distance preceding context. Then, a novel word-based approach [12, 24] was proposed to directly model candidate segmented results. Unfortunately, both of them applied pre-processing steps which either requires outer resources or alters the testset. Zhou et al. [38] propose a simple method to train char embeddings on auto labeled data, which brings significantly improvement. Yang et al. [37] demonstrates the effectiveness of rich pre-training on external resources. With the help of contextualized embeddings like Bidirectional Encoder Representations (BERT) [32], much progress has been made in the last 2 years toward delivery of high accuracy CWS models. Huang et al. [30] propose a domain adaptive segmenter to exploit diverse criteria datasets, which delivers the state-of-the-art accuracy. Qiu et al. [34] and Ke et al. [36] encode each dataset [35] with the unified schema proposed in [35], largely reducing number of decoders. Tian et al. [31] incorporate wordhood information with BERT, demonstrating the robustness on smaller datasets. Their work differ from us in that they focus on pushing the accuracy at the cost of runtime speed, while we explore a method to balance both accuracy and speed.

3. Approach

3.1 Word segmentation as sequence labeling

To treat CWS as a problem of character-based sequence tagging is common and effective. One widely used tagging set is $\mathcal{T} = \{B, M, E, S\}$, which respectively represents for the **b**egin, **m**iddle, **e**nd of a word, or **s**ingle character which forms a word. Then the corpus is converted to a sequence of characters along with their tags. Upon which a general tagging model is trained. In testing phase, the segmentation decision is made once the label sequence is predicted by the tagging model. We will explain our sequence tagging models in this section.

3.1.1 Conditional probability sequence tagging

Given a sequence $\mathbf{X}$ with n characters as $\mathbf{X} = (\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n)$, the goal of sequence tagging on CWS is to find the most possible tags $\mathbf{Y}^* = \{\mathbf{y}_1^*, \ldots, \mathbf{y}_n^*\}$:

$$\mathbf{Y}^* = \arg\max_{\mathbf{Y} \in \mathcal{T}^n} P(\mathbf{Y}|\mathbf{X}), \tag{1}$$

where $\mathcal{T} = \{B, M, E, S\}$.

Many sequence labeling models can be applied to this task. For example, [2] applied Maximum Entropy Markov Model (MEMM), then the CRFs [3,4,14,16–18] quickly took the charge. We also apply CRFs to capture interactions between adjacent labels.

Next, we consider two factorizations of this conditional distribution depending on whether decisions are made independently or not.

3.1.2 Conditionally independent inference

Given features representation $\mathbf{h}_t$ for character $\mathbf{x_t}$, conditionally independent probability is defined as:

$$P(\mathbf{Y}|\mathbf{X}) = \prod_{t=1}^{T} P(\mathbf{y}_t | f(\mathbf{x}_t)), \tag{2}$$

where $\mathbf{h}_t = f(\mathbf{x}_t)$ is the feature function.

As decisions are independent, prediction has linear time complexity $O(n)$ and can be parallelized across the whole sentence:

$$\mathbf{y}_t^* = \arg\max_{\mathbf{y}_t \in \mathcal{T}} P(\mathbf{y}_t | f(\mathbf{x}_t)),$$
$$\mathbf{Y}^* = (\mathbf{y}_1^*, \mathbf{y}_2^*, \ldots, \mathbf{y}_n^*).$$

3.1.3 Linear chain CRF inference

Strong dependencies across output tags are generated by the definition of tagging set. For instance, **b**egin can't follow **m**iddle or **s**ingle. Conditionally independent

classification models are unaware of this interaction, leading to invalid label sequence. Therefore, a more appealing solution is to model labels jointly using a Markov chain, or more concretely linear-chain CRFs [3].

Given contextual features representation $\mathbf{h}_t$ for character $\mathbf{x_t}$, CRFs firstly use a linear score function $s(\mathbf{X}, t) \in \mathbb{R}^{|\mathcal{T}|}$ to generate a local score for each tag:

$$s(\mathbf{X}, t) = \mathbf{W}_s^\top \mathbf{h}_t + \mathbf{b}_s,$$

where $\mathbf{W}_s \in \mathbb{R}^{d_h \times |\mathcal{T}|}$ and $\mathbf{b}_s \in \mathbb{R}^{|\mathcal{T}|}$ are trainable parameters.

Then, for a sequence of predictions:

$$\mathbf{Y} = (y_1, y_2, \ldots, y_n),$$

first order linear-chain CRFs employ a Markov chain to compute its global score as:

$$s(\mathbf{X}, \mathbf{Y}) = \sum_{i=0}^{n} A_{y_i, y_{i+1}} + \sum_{i=1}^{n} P_{i, y_i},$$

where $\mathbf{A}$ is a transition matrix. $A_{i,j}$ represents for the score of a transition from the tag i to tag j. y_0 and y_n are the *start* and *end* tag of a sentence. $\mathbf{A}$ is therefore a square matrix of size $|\mathcal{T}| + 2$.

Finally, this global score is normalized to a probability in Eq. (1) via a softmax over all possible tag sequences:

$$P(\mathbf{Y}|\mathbf{X}) = \frac{e^{s(\mathbf{X}, \mathbf{Y})}}{\sum_{\widetilde{\mathbf{Y}} \in \mathbf{Y_X}} e^{s(\mathbf{X}, \widetilde{\mathbf{Y}})}}.$$

As first order linear chain CRFs only model bigram interactions between output tags, so the maximum posteriori probability of label sequence $\mathbf{Y}^*$ in Eq. 1 can be computed using dynamic programming, both in training and decoding phase. This inference process is illustrated as Fig. 1.

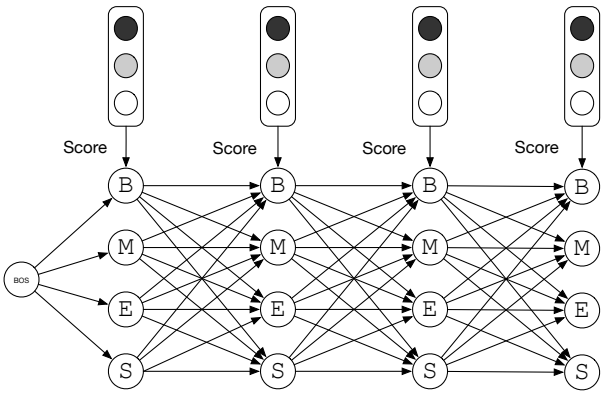

Fig. 1 *CRF layer.*

3.2 Neural feature extractors

Regarding the extraction of contextual features representation $\mathbf{h}_t$, two prevailing approaches for neural feature extraction, Recurrent Neural Networks (RNNs) and Convolutional Neural Networks (CNNs), are studied in this work.

3.2.1 Bi-LSTM feature extraction

Bi-LSTM Given a sentence $(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n)$ consisted of n characters, in which each character is represented as a d-dimensional vector, Long Short-Term Memory Networks (LSTMs) can only produce one representation $\overrightarrow{\mathbf{h}_t}$ for the left context at every character t. For generating the missing representation of the right context $\overleftarrow{\mathbf{h}_t}$, a second LSTM reads the same sequence but in reverse order. One pair of forward and backward LSTM is called bidirectional LSTM (Bi-LSTM) [19] in literature. By concatenating its left and right context representations, the final representation is produced as $\mathbf{h}_t = [\overrightarrow{\mathbf{h}_t}; \overleftarrow{\mathbf{h}_t}]$.

The motivation behind extracting feature via Bi-LSTM is that, segmentation decision depends on nonlocal context information from both preceding and succeeding characters. In early sequence model, a sliding window centered in current character is quite common for feature extracting. Now Bi-LSTM hits the mainstream. Our architecture for contextual feature capturing is shown in Fig. 2. This contextual feature vector encodes both the meaning of a character and its context.

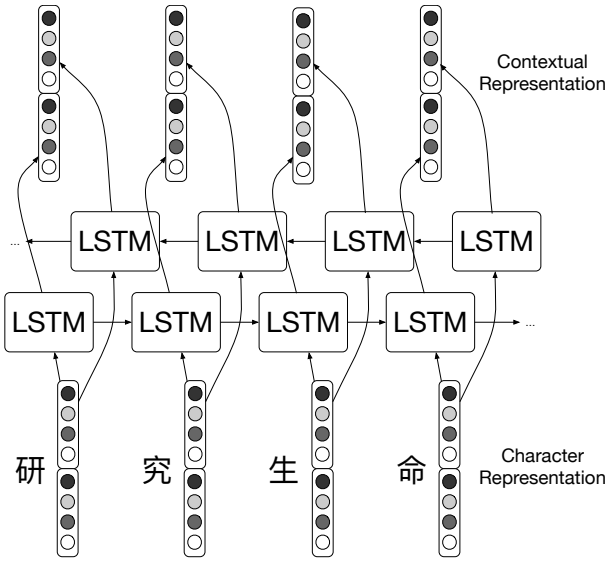

Fig. 2 *Character LSTM layer.*

3.2.2 Iterated dilated CNNs feature extraction

In computer vision, a pixel is typically a vector with 3 channels: red, green and blue (RGB). At meanwhile, one picture has two dimensions of pixels. In the case

of Nature Language Processing (NLP), a word is represented by a vector with hundreds of channels, whereas a sentence has only one dimension of words. As a result, convolution filters [9] in NLP are typically one-dimensional vectors, applied to a sequence of word vectors, or more precisely a matrix of the same width with filters. Formally, output c_t of convolutional operator applied to each word $\mathbf{x}_t$ is defined as:

$$\mathbf{c}_t = \mathbf{W_c} \bigoplus_{k=0}^{r} \mathbf{x}_{t \pm k}, \tag{3}$$

where $\oplus$ is vector concatenation, $2r+1$ is the filter width w. Therefore, the effective input width of every token is limited by w.

To enlarge effective input width, dilated convolution [20] transforms one token every δ input, where δ is the dilation width. The dilated convolution operator is defined as:

$$\mathbf{c}_t = \mathbf{W_c} \bigoplus_{k=0}^{r} \mathbf{x}_{t \pm k\delta}. \tag{4}$$

In this way, dilated convolution with $\delta > 1$ can incorporate broader context into the contextual representation than naive convolution, without increasing the amount of parameters. To further capture global context, ID-CNN stacks multiple dilated convolutions bottom up, then feeds the outputs from one filter to the next.

Iterated Dilated CNNs Although increasing the layers of stacked dilated convolutions can help to capture global context, it in turn introduces more parameters, leading to over-fitting on small datasets. Strubell et al. [11] presented a method to reuse the same filter but apply different δ across layers. In its recursive pattern, parameters amount remains constant while network goes deeper and deeper. This variant is referred as iterated dilated CNNs (ID-CNNs).

We applied iterated dilated CNN as a fast feature detector for CWS. The layer architecture is shown in Fig. 3.

3.3 Training

The training procedure is to maximize the log-probability of the gold tag sequence. Depending on wether it makes independent decision or not, the probabilistic models in Section 2 are named as non-sequential or sequential model.

3.3.1 Non-sequential model

For conditionally independent model, the log-probability is:

$$\log(P(\mathbf{Y}|\mathbf{X})) = \sum_{t=1}^{n} \log P(\mathbf{y_t} \mid \mathbf{h_t}).$$

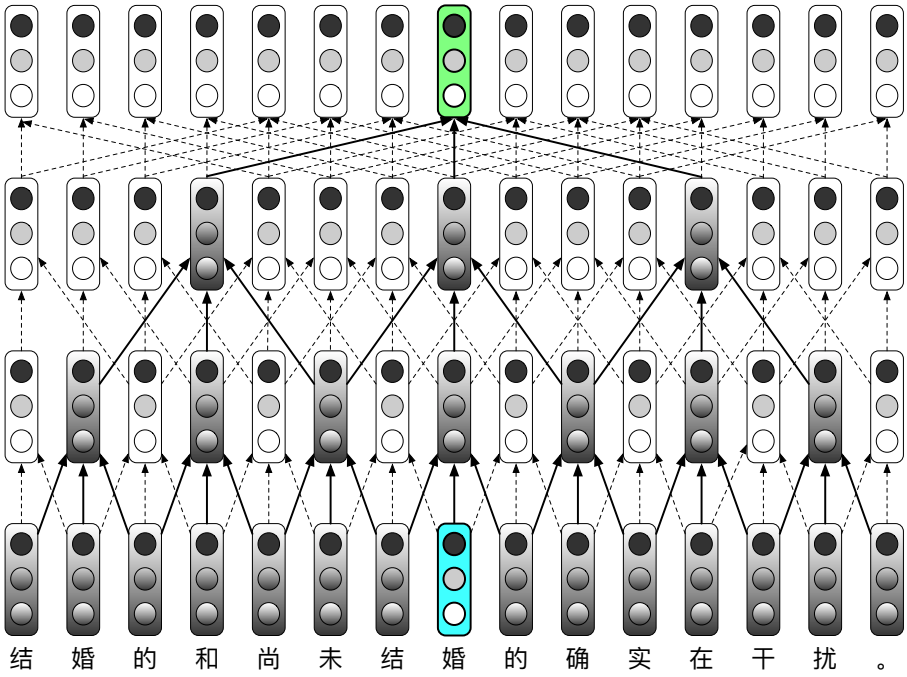

Fig. 3 *Iterated Dilated CNN with $w = 3$ and $\delta = 4$. Current token, effective inputs and final contextual representation are highlighted.*

3.3.2 Sequential model

For CRF model, this is:

$$\begin{aligned}\log(P(\mathbf{Y}|\mathbf{X})) &= s(\mathbf{X}, \mathbf{Y}) - \log\left(\sum_{\widetilde{\mathbf{Y}} \in \mathbf{Y_X}} e^{s(\mathbf{X}, \widetilde{\mathbf{Y}})}\right) \\ &= s(\mathbf{X}, \mathbf{Y}) - \underset{\widetilde{\mathbf{Y}} \in \mathbf{Y_X}}{\text{logadd}}\ s(\mathbf{X}, \widetilde{\mathbf{Y}}),\end{aligned}$$

where $\mathbf{Y_X}$ represents all possible tag sequences for a sentence $\mathbf{X}$. As only bigram interactions between tags are modeled, we can compute the above summation by dynamic programming.

4. Experiments

We conducted two parts of experiments to compare performance and speed of the following models:

1. Bi-LSTM models with or without CRF layer.
2. ID-CNN models with or without CRF layer.

4.1 Datasets

To explore these questions, we experimented on the 4 prevalent CWS datasets from SIGHAN2005 [21], as these datasets are commonly available and used by previous state-of-the-art works. Following conventions, the last 10 % sentences of training set are used as development set. All datasets are preprocessed by replacing the continuous English characters and digits with a unique token. Tab. I describes statistics of the Sighan 2005 datasets used in our experiments.

	Corpora		#words	#chars#	word types	char types	OOV
Sighan05	MSRA	Train	2.4M	4.0M	75.4K	5.1K	
		Test	0.1M	0.2M	11.9K	2.8K	1.32 %
	AS	Train	5.4M	8.3M	128.8K	5.8K	
		Test	0.1M	0.2M	18.0K	3.4K	2.20 %
	PKU	Train	1.1M	1.8M	51.2K	4.6K	
		Test	0.1M	0.2M	12.5K	2.9K	2.06 %
	CITYU	Train	1.1M	1.8M	43.4K	4.2K	
		Test	0.2M	0.4M	23.2K	3.6K	3.69 %

Tab. I *Details of the 4 datasets in our experiments. "OOV" is Out-Of-Vobulary rate.*

4.2 Pre-training

The corpus used for pre-training is Chinese Wikipedia dump of July 2017. Traditional Chinese characters inside are converted to Simplified Chinese via the popular Chinese NLP tool HanLP[3].

Instead of the commonly used word2vec [22], we utilized fastText [23][4] to train character embeddings. We chose SG model, 100 dimension, and set the initial learning rate to 0.1.

4.3 Hyper parameters

Hyper parameters are tuned on development sets. Some crucial common hyper parameters shared by all models are listed in Tab. II.

4.4 Implementation details

We implement our model in TensorFlow [33] and conduct all experiments on a GeForce GTX TITAN GPU. Our codes are written in Python and requires a Linux environment to run. Depending on model architecture and dataset size, each experiment setting takes about 1 to 4 hours to train on our GPU.

[3] https://github.com/hankcs/HanLP
[4] https://github.com/facebookresearch/fastText

Radical embedding dim	$d_r = 50$
Character embedding dim	$d_c = 100$
L2 regularization	$l2 = 10^{-5}$
Batch size	$b = 128$
Learning rate	$e_0 = 0.001$
Max epochs	$n = 100$

Tab. II *Model settings*

.

4.5 Results on SIGHAN bakeoff 2005

Four models are trained and tested on four datasets respectively. The final results are shown in Tab. III.

Models	PKU	MSR	CityU	AS
Chen et al. [13]♠	94.5	95.4	–	–
Chen et al. [8]♠	94.8	95.6	–	–
Chen et al. [16]	94.3	96.0	**95.6**	94.8
Cai et al. [12]♢	**95.8**	97.1	**95.6**	**95.3**
Zhou et al. [38]‡	–	–	96.0	97.8
Yang et al. [37]‡	96.3	97.5	95.7	96.9
Huang et al. [30]†‡	96.6	97.9	97.6	96.6
Tian et al. [31]‡	96.5	98.4	97.9	96.6
Qiu et al. [34]†	96.4	98.1	96.9	96.4
Ke et al. [36]†‡	96.9	98.5	97.1	96.9
Bi-LSTM	94.3	95.7	94.6	95.0
Bi-LSTM-CRF	94.8	96.7	95.5	**95.3**
ID-CNN	95.2	96.7	**95.6**	95.1
ID-CNN-CRF	95.2	97.0	95.5	95.0

Tab. III *Comparison with state-of-the-art models of results on all four Bakeoff-2005 datasets. Multi-Criteria Learning*† *and BERT/silver data pre-training*‡ *are orthogonal to our work.*

Results with ♣ used external dictionary or corpus. Results with ♠ are from Cai and Zhao [24]'s running on their released implementations without dictionary. Results with ♢ expurgated long words in test set. Note that PKU, CityU used by Chen et al. [16] are from SIGHAN08, while others' are from SIGHAN05. The scores from recent works [30, 31, 34, 36–38], which either or both use more training data or adopt external large-scale pre-trained BERT, are also included for an indirect comparison.

We also recorded the training speed of every model on a GeForce GTX TITAN GPU in Tab. IV. The last column shows the relative training speed of neural models without BERT. We are unable to experiment with the BERT models‡ due to their

Model	Speed
Adversarial NN [16]	1.00×
GRU [13]	3.21×
LSTM [8]	6.73×
GCNN [12]	8.52×
Bi-LSTM-CRF	5.60×
Bi-LSTM	6.55×
ID-CNN-CRF	12.76×
ID-CNN	19.48×

Tab. IV *Relative training-time speed of tagging models on AS dataset.*

requirement of computation resource. The BERT encoder itself is approximately 50 times larger than our largest model, which exceeds our GPU memory constraint. We anticipate them to be approximately one or two orders of magnitude slower than our models, based on the speed reported by Huang et al. [30]. Non-neural models are not comparable since they are not GPU-optimized. The speed and performance of our LSTM baseline model are very close to its LSTM counterpart [8]. Although decoding speed usually drops when the model complexity increases, our models are less sensitive as our improvement is mostly made on the data part instead of model structure. After integration of CRF layer, the speed of our model slightly decreases while still outperforms their models [12] of similar performance.

4.6 Analysis

ID-CNN has comparable or even better performance over LSTM With or without CRF layer, ID-CNN models outperformed their BiLSTM competitors. On MSR and CityU datasets, ID-CNN repeatedly showed a dramatic 1 % advantage over Bi-LSTM. Bi-LSTM is capable of capturing long term dependencies, while the length of Chinese words are usually less than 2 characters. The dependency on characters far away is less important than that of neighbor characters. Instead, ID-CNN models the n-gram within a sliding window, which is enough for making tokenization decisions without degrading the runtime speed.

ID-CNN has faster speed over LSTM With or without CRF layer, ID-CNN models are 2× faster than their BiLSTM competitors, with no accuracy loss or even better performance.LSTM suffers from dependency on outputs from previous characters, where parallelization is impossible. ID-CNN only depends on previous layer and number of layers are much smaller than number of characters in a sentence.

BiLSTM relies heavily on CRF layer Without CRF layer, BiLSTM suffers an accuracy loss up to 1 %. This unavoidable dependency on dynamic search leads to an incompatibility with parallelization. CRF layer helps to penalize invalid transitions of tags, which is important for sequence labeling task such as word

segmentation. Without CRF, LSTM tends to over estimate the importance of long term dependency without global optimization, which results to significant performance drop.

CRF layer does tiny enhancement to tagging inference for ID-CNN Without CRF layer, ID-CNN's tagging inference is still accurate, with a loss of accuracy between 0 to 0.3 %. We also noticed that on CityU dataset, ID-CNN performs slightly better than ID-CNN-CRF, which further reflects the accuracy of ID-CNN. Generally, when individual decision is accurate, transition feature used in CRF will be less important. By removing the dependency of CRF layer, parameters are reduced, meanwhile ID-CNN can take more advantage of parallelization.

5. Conclusions and future works

5.1 Conclusions

In this paper, we have proposed a novel approach to replace BiLSTM with ID-CNN in CWS systems. Our extensive experiments showed that ID-CNN has great advantages in runtime speed while keeping the same or producing even better accuracy. We also showed that ID-CNN relies less on CRF layer, thus it can generate more accurate results solely on local features, which is in turn more appropriate for parallelization.

5.2 Future works

The radical embeddings used by ID-CNN are trained with non-contextualized methods, which could result in loss of poly-semantic information. The same radical could have multiple meaning in different context, while our current approach assigns unique embedding to it. We are looking forward to incorporating recent contextualized methods with our radical embeddings.

Our ID-CNN sequence tagging framework can be applied to extensive NLP tasks like Part-of-Speech tagging and Named Entity Recognition (NER). These tasks will benefit from faster runtime speed without performance loss. We have made our work publicly available and encourage more applications of this promising model.

References

[1] NIANWEN X. Chinese Word Segmentation as Character Tagging. International Journal of Computational Linguistics & Chinese Language Processing (IJCLCLP), 2003, 8(1), pp. 29–48.

[2] KIAT L.J., HWEE TOU N., WENYUAN G. A Maximum Entropy Approach to Chinese Word Segmentation. Proceedings of the Fourth Sighan Workshop on Chinese Language Processing, 2005.

[3] LAFFERTY J.D., MCCALLUM A., PEREIRA F.C.N. Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data. In: *Eighteenth International Conference on Machine Learning*, 2001, pp. 282–289, doi: 10.5555/645530.655813.

[4] PENG F., FENG F., MCCALLUM A. Chinese Segmentation and New Word Detection using Conditional Random Fields. Proceedings of the 20th International Conference on Computational Linguistics, pp. 562–568, 2004.

[5] ZHENG X., CHEN H., XU T. Deep Learning for Chinese Word Segmentation and POS Tagging. Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing (EMNLP), pp. 647–657, 2013.

[6] COLLOBERT R., WESTON J., BOTTOU L., KARLEN M., KAVUKCUOGLU K., KUKSA P.P. Natural Language Processing (Almost) from Scratch. *Journal of Machine Learning Research*, 2011. doi: arXiv:1103.0398.

[7] PEI W., GE T., CHANG B. Max-Margin Tensor Neural Network for Chinese Word Segmentation. *ACL*, 2014, doi: 10.3115/v1/P14-1028.

[8] CHEN X., QIU X., ZHU C., LIU P., HUANG X. Long Short-Term Memory Neural Networks for Chinese Word Segmentation. Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP), 2015, pp. 1197–1206, doi: 10.18653/v1/D15-1141.

[9] YOON K. Convolutional neural networks for sentence classification, Empirical methods in natural language processing. doi: arXiv:1408.5882.

[10] KALCHBRENNER N., GREFENSTETTE E., BLUNSOM P. A convolutional neural network for modelling sentences, Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, 2014, pp. 655–665. doi: arXiv:1404.2188.

[11] STRUBELL E., VERGA P., BELANGER D., MCCALLUM A. Fast and Accurate Entity Recognition with Iterated Dilated Convolutions, Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pp. 2660–2670. doi: arXiv:1702.02098.

[12] CAI D., ZHAO H., ZHANG Z., XIN Y., WU Y., HUANG F. Fast and Accurate Neural Word Segmentation for Chinese. *arXiv.org*, 2017, doi: 10.18653/v1/P17-2096.

[13] CHEN X., QIU X., ZHU C., HUANG X. Gated Recursive Neural Network for Chinese Word Segmentation. Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing, 2015, 1, pp. 1744–1753, doi: 10.3115/v1/P15-1168.

[14] DONG C., ZHANG J., ZONG C., HATTORI M., DI H. Character-Based LSTM-CRF with Radical-Level Features for Chinese Named Entity Recognition. Natural Language Understanding and Intelligent Applications. ICCPOL 2016, NLPCC 2016. Lecture Notes in Computer Science, 10102, 2016. doi: 10.1007/978-3-319-50496-4_20.

[15] ZHANG L., WANG H., SUN X., MANSUR M. Exploring representations from unlabeled data with co-training for Chinese Word Segmentation. In: *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, Seattle, Washington, USA, October 2013, pp. 311–321.

[16] CHEN X., SHI Z., QIU X., HUANG X. Adversarial Multi-Criteria Learning for Chinese Word Segmentation. Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, 2017, 1, pp. 1193–1203, doi: 10.18653/v1/P17-1110.

[17] TSENG H., CHANG P., ANDREW G., JURAFSKY D., MANNING C. A Conditional Random Field Word Segmenter for Sighan Bakeoff 2005. Proceedings of the Fourth SIGHAN Workshop on Chinese Language Processing, 2005, pp. 168–171.

[18] ZHAO H., HUANG C., LI M., LU B.-L. Effective Tag Set Selection in Chinese Word Segmentation via Conditional Random Field Modeling. Proceedings of the 20th Pacific Asia Conference on Language, Information and Computation (PACLIC), 2006, pp. 87–94.

[19] GRAVES A., SCHMIDHUBER J. Framewise Phoneme Classification with Bidirectional LSTM and Other Neural Network Architectures. Neural Networks, 2005, 18(5-6), pp. 602–610, doi: 10.1016/j.neunet.2005.06.042.

[20] YU F., KOLTUN V. Multi-scale context aggregation by dilated convolutions, Intl. Conference on Learning Representations (ICLR). 2015.

[21] EMERSON T. The Second International Chinese Word Segmentation Bakeoff. In Proceedings of the Fourth SIGHAN Workshop on Chinese Language Processing, pp. 123–133. Jeju Island, Korea, 2005.

[22] MIKOLOV T., CHEN K., CORRADO G., DEAN J. Efficient Estimation of Word Representations in Vector Space, arXiv.org, 1301.3781v3, arxiv, cs.CL, 2013.

[23] BOJANOWSKI P., GRAVE E., JOULIN A., MIKOLOV T. Enriching Word Vectors with Subword Information. *arXiv.org*, 2016.

[24] CAI D., ZHAO H. Neural Word Segmentation Learning for Chinese. *arXiv.org*, 2016.

[25] ZHANG Y., CLARK S. Chinese segmentation with a word-based perceptron algorithm. Association for Computational Linguistics, Prague, Czech Republic, 2007, pp. 840–847.

[26] SUN X., ZHANG Y., MATSUZAKI T., TSURUOKA Y., TSUJII J. A Discriminative Latent Variable Chinese Segmenter with Hybrid Word/Character Information. Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics, 2009, pp. 56–64.

[27] SUN X., WANG H., LI W. Fast Online Training with Frequency-Adaptive Learning Rates for Chinese Word Segmentation and New Word Detection. Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics, 2012, 1, pp. 253–262.

[28] ZHAO H., CAI D., HUANG C., KIT C. Chinese word segmentation: A decade review, Journal of Chinese Information Processing, 21(3), pp. 8–19.

[29] ZHAO H., HUANG C.N., LI M., LU B.L. A Unified Character-Based Tagging Framework for Chinese Word Segmentation, ACM Trans. on Asian Language Information Processing, 9(2), pp. 1–32. doi: 10.1145/1781134.1781135.

[30] HUANG W., CHENG X., CHEN K., WANG T., CHU W. Toward fast and accurate neural chinese word segmentation with multi-criteria learning. *arXiv preprint arXiv:1903.04190*, 2019.

[31] TIAN Y., SONG Y., XIA F., ZHANG T., WANG Y. Improving Chinese word segmentation with wordhood memory networks. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, Online. Association for Computational Linguistics, 2020, pp. 8274–8285, https://doi.org/10.18653/v1/2020.acl-main.734.

[32] DEVLIN J., CHANG M.W., LEE K., TOUTANOVA K. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 4171–4186, 2019, https://www.aclweb.org/anthology/N19-1423.

[33] ABADI M., BARHAM P., CHEN J., CHEN Z., DAVIS A., DEAN J., DEVIN M., GHEMAWAT S., IRVING G., ISARD M. Tensorflow: A system for large-scale machine learning. In *12th {USENIX} symposium on operating systems design and implementation ({OSDI} 16)*, 2016, pp. 265–283.

[34] QIU X., PEI H., YAN H., HUANG X. Multi-criteria chinese word segmentation with transformer, 2019, *arXiv preprint arXiv:1906.12035*.

[35] HE H., WU L., YAN H., GAO Z., FENG Y., TOWNSEND G. Effective neural solution for multi-criteria word segmentation. In: *Smart Intelligent Computing and Applications*, 2019, pp. 133–142.

[36] KE Z., SHI L., MENG E., WANG B., QIU X., HUANG X. Unified multi-criteria chinese word segmentation with bert, 2020, *arXiv preprint arXiv:2004.05808*.

[37] YANG J., ZHANG Y., DONG F. Neural word segmentation with rich pretraining. In: *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Vancouver, Canada. Association for Computational Linguistics, 2017, pp. 839–849, doi: 10.18653/v1/P17-1078.

[38] ZHOU H., YU Z., ZHANG Y., HUANG S., DAI X., CHEN J. Word-context character embeddings for Chinese word segmentation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, Copenhagen, Denmark. Association for Computational Linguistics, 2017, pp. 760–766, doi: 10.18653/v1/D17-1079.