Improved Differentiable Architecture Search for Language Modeling and Named Entity Recognition

Yufan Jiang[1], **Chi Hu**[1], **Tong Xiao**[12]*, **Chunliang Zhang**[12], **Jingbo Zhu**[12]
[1]NLP Lab, Northeastern University, Shenyang, China
[2]NiuTrans Research, Shenyang, China
jiangyufan2018@outlook.com, huchinlp@gmail.com
{xiaotong, zhangchunliang, zhujingbo}@mail.neu.edu.cn

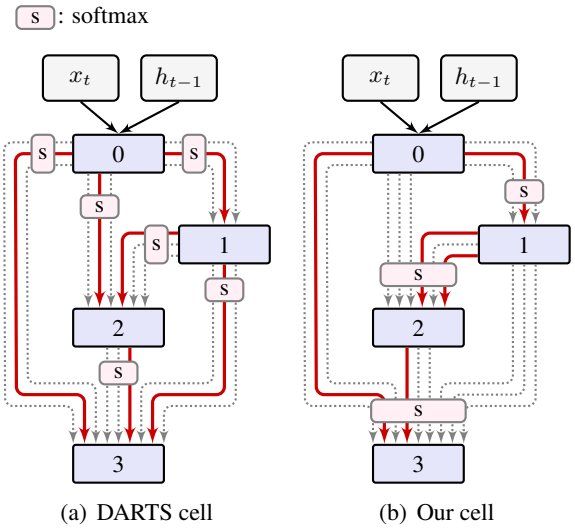
Abstract

In this paper, we study differentiable neural architecture search (NAS) methods for natural language processing. In particular, we improve differentiable architecture search by removing the softmax-local constraint. Also, we apply differentiable NAS to named entity recognition (NER). It is the first time that differentiable NAS methods are adopted in NLP tasks other than language modeling. On both the PTB language modeling and CoNLL-2003 English NER data, our method outperforms strong baselines. It achieves a new state-of-the-art on the NER task.

1 Introduction

Neural architecture search (NAS) has become popular recently in machine learning for their ability to find new models and to free researchers from the hard work of designing network architectures. The earliest of these approaches use reinforcement learning (RL) to learn promising architectures in a discrete space (Zoph and Le, 2016), whereas others have successfully modeled the problem in a continuous manner (Liu et al., 2019; Xie et al., 2019b; Huang and Xiang, 2019). As an instance of the latter, differentiable architecture search (DARTS) employs continuous relaxation to architecture representation and makes gradient descent straightforwardly applicable to search. This leads to an efficient search process that is orders of magnitude faster than the RL-based counterparts.

Like recent methods in NAS (Xie and Yuille, 2017; Zoph and Le, 2016; Baker et al., 2016), DARTS represents networks as a directed acyclic graph for a given computation cell (see Figure 1(a)). An edge between nodes performs a predefined operation to transform the input (i.e., tail) to the output (i.e., head). For a continuous network space, DARTS uses the softmax trick to relax the categorical choice of edges to soft decisions. Then, one can optimize over the graph using standard gradient descent methods. The optimized network is inferred by choosing the edges with maximum weights in softmax.



(a) DARTS cell (b) Our cell

Figure 1: An overview of DARTS cell and our cell

However, DARTS is a "local" model because the softmax-based relaxation is imposed on each bundle of edges between two nodes. This leads to a biased model in that edges coming from different nodes are not comparable. Such a constraint limits the inference space to sub-graphs with one edge between each pair of nodes. Also, the learned network might be redundant because every node has to receive edges from all predecessors no matter they are necessary or not. This problem is similar to the bias problem in other graph-based models where local decisions make the model non-optimal (Lafferty et al., 2001; Daphne Koller and Nir Friedman, 2009).

Here we present an improvement of DARTS,

*Corresponding author.

Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, pages 3583–3588, Hong Kong, China, November 3–7, 2019.

called I-DARTS, that further relaxes the softmax-local constraint. The idea is simple - we consider all incoming edges to a given node in a single softmax. This offers a broader choice of edges and enlarges the space we infer the network from. For example, one can simultaneously select multiple important edges between two nodes and leave some node pairs unlinked (see Figure 1(b)).

I-DARTS outperforms strong baselines on the PTB language modeling and CoNLL named entity recognition (NER) tasks. This gives a new state-of-the-art on the NER dataset. To our knowledge, it is the first time to apply differentiable architecture search methods to NLP tasks other than language modeling. More interestingly, we observe that our method is 1.4X faster than DARTS for convergence of architecture search. Also, we provide the architectures learned by I-DARTS, which can be referred for related tasks.

2 The Method

Although we will restrict ourselves to language modeling and NER for experiments, in the section, we discuss the more general case. We choose recurrent neural networks (RNNs) to model the sequence generation and tagging problems. Given a sequence of input vectors $\{x_1, ..., x_L\}$, we repeat applying RNN cells to generate the output $\{h_1, ..., h_L\}$. The RNN cell is defined as: $h_t = g(x_t, h_{t-1})$, where t is the time step and $g(\cdot, \cdot)$ is the function of the cell. In NAS, the objective is to search for a good $g(\cdot, \cdot)$ in an automatic fashion.

2.1 Architecture Search Space

We follow the assumption that $g(\cdot, \cdot)$ is a DAG consisting of N nodes and edges among them (Liu et al., 2019; Xie et al., 2019b; Pham et al., 2018). An edge $o^{i,j}(\cdot)$ between node pair (i, j) indicates an activation function from node j to node i. For node i, it simply sums over vectors from all predecessor nodes ($j < i$), followed by a linear transformation with a parameter matrix W_i. More formally, let s_i be the state of node i. We define s_i to be:

$$s_i = \sum_{j<i} o^{i,j}(s_j \cdot W_j) \tag{1}$$

See Figure 1 for an example network (red lines). Note that this model can encode an exponential number of graphs by choosing different sub-sets of edges (i.e., choosing $o^{i,j}(\cdot)$ for each (i,j)). The output of search is the optimal edge selection and the corresponding network.

2.2 Improved DARTS

Given a set of edges $\{o_k^{i,j}\}$, one can try each $o_k^{i,j}$ to induce a network, and then train and evaluate it. The optimal choice is the edge with highest accuracy on the validation set. In I-DARTS, we instead do this in a soft way. We re-define s_i as:

$$s_i = \sum_{j<i} \sum_k \alpha_k^{i,j} \cdot o_k^{i,j}(s_j \cdot W_j) \tag{2}$$

where $\alpha_k^{i,j}$ is the weight indicating the importance of $o_k^{i,j}(\cdot)$. It is computed by the softmax normalization over edges between nodes i and j, like this

$$\alpha_k^{i,j} = \frac{\exp(w_k^{i,j})}{\sum_{k'} \exp(w_{k'}^{i,j})} \tag{3}$$

where $w_k^{i,j}$ is the model parameter. This model reduces the architecture search problem to learn continuous variables $\{\alpha_k^{i,j}\}$, which can be implemented using efficient gradient descent methods. After training, the final architecture is encoded by the edges with largest weights.

Eq. (3) imposes a constraint that weights $\{\alpha_k^{i,j}\}$ are normalized for each j. Such a model in general faces the local decision and bias problems as pointed out in graph-based methods (Lafferty et al., 2001; Daphne Koller and Nir Friedman, 2009). Moreover, the inference has to be performed in a smaller space because we have to infer exactly one edge between each node pair and exclude networks violating this constraint.

Here we remove the constraint and system bias. To this end, we compute the softmax normalization over all incoming edges for node i:

$$\alpha_k^{i,j} = \frac{\exp(w_k^{i,j})}{\sum_{j<i} \sum_{k'} \exp(w_{k'}^{i,j})} \tag{4}$$

It provides us a way to compare all incoming edges in the same manner, rather than making a local decision via a bundle of edges from node j. As another bonus, this method can search for networks that are not covered by DARTS, e.g., networks that contain two edges between the same node pair.

See Figure 1(b) for an illustration of our method. To infer the optimal architecture, we basically do the same thing as in DARTS. The differ-

ence lies in that we select top-n edges with respect to $\alpha_k^{i,j}$. Here n is a hyper-parameter that controls the density of the network. E.g., $n = 1$ means a sparse net, and $n = \infty$ means a very dense net involving all those edges.

3 Experiments

We test our method on language modeling and named entity recognition tasks. Our experiments consist of two parts: recurrent neural architecture search and architecture evaluation. In architecture search, we search for good RNN cell architectures. Then, we train and evaluate the learned architecture.

3.1 Architecture Search

For language modeling, we run neural search on the PTB corpus. We use the standard preprocessed version of the dataset (Pham et al., 2018). To make it comparable with previous work, we copy the setup used in (Pham et al., 2018; Liu et al., 2019). The recurrent cell consist of 8 nodes. The candidate operation set of every edge contain 5 activation functions, including zeroize, tanh, relu, sigmoid, and identity. To learn architectures, we run the search system for 40 training epochs with a batch size of 256. We optimize models parameters $\{W_i\}$ using SGD with a learning rate of 20 and a weight decay rate of 5e-7, and optimized softmax relaxation parameters $\{w_k^{i,j}\}$ by Adam with a learning rate of 3e-3 and a weight decay rate of 1e-3. For RNN models, we use a single-layer recurrent network with embedding and hidden sizes = 300. It takes us 4 hours to learn the architecture on a single GPU of NVIDIA 1080Ti.

For named entity recognition, we choose the CONLL-2003 English dataset. We follow the same setup as in language modeling but with a different learning rate (0.1) and a different hidden layer size (256). It takes us 4 hours to learn the architecture on the same GPU.

3.2 Architecture Evaluation

Firstly, the discovered architecture is evaluated on the language modeling task. Before that, we train it on the same data used in architecture search. The size of hidden layers is set to 850. We use averaged SGD to train the model for 3,000 epochs, with a learning rate of 20 and a weight decay rate of 8e-7. For a fair comparison, we do not fine-tune the model at the end of the training.

Architecture	Perplexity		Search Cost
	val	test	(GPU days)
V-RHN	67.9	65.4	-
LSTM	60.7	58.8	-
LSTM + SC	60.9	58.3	-
LSTM + SE	58.1	56.0	-
ENAS	60.8	58.6	0.50
DARTS	58.3	56.1	0.25
Random RNNs	63.7	61.2	-
I-DARTS ($n = 1$)	58.0	56.0	0.17
I-DARTS ($n = 2$)	-	-	-

Table 1: Perplexities on PTB (lower is better). V-RHN (Zilly et al., 2016) indicates Variational RHN. LSTM + SC (Yang et al., 2018) indicates LSTM with skip connection. LSTM + SE (Merity et al., 2018) indicates LSTM with mixture of softmax. Random RNNs indicates that the network generated by random initialized.

Table 1 shows the perplexities of different RNN models on PTB. We also report the results of previous systems. The model discovered by I-DARTS achieves a validation perplexity of 58.0 and a test perplexity of 56.0 when $n = 1$. It is on par with the state-of-the-art models that are designed either manually or automatically. However, we find that the model failed to optimize when $n = 2$. It might result from the complex interaction between operations. We leave this issue for future study.

Since architecture search is initialization-sensitive (Pham et al., 2018; Liu et al., 2019), we search the architectures for 4 times with different random seeds. We evaluate the architecture every 10 search epochs by retraining it on PTB for 500 epochs. We compare DARTS with our I-DARTS method with the same random seed. See Figure 2(b) for averaged validation perplexities over 4 different runs at different search epochs. We see that I-DARTS is easier to converge than DARTS (4 hours). It is 1.4X faster than that of DARTS. Another interesting finding is that I-DARTS achieves a lower validation perplexity than DARTS during architecture search. This may indicate better architectures found by I-DARTS because the search model is optimized with respect to validation perplexity.

Then, we test the learned architecture in a named entity recognition system on the English data from CoNLL-2003 shared task (Sang and Meulder, 2003). Following previous work (Akbik et al., 2018; Peters et al., 2017), we report the averaged F1 score over 5 runs on the test set. For modeling, we choose the single-layer RNN-CRF

model because it achieved state-of-the-art results on several sequence labeling tasks (Lample et al., 2016; Ma and Hovy, 2016). We use GloVe 100-dimensional word embeddings (Pennington et al., 2014) and pooled contextual embeddings (Akbik et al., 2019) as pre-trained word embeddings. We replace the standard bidirectional LSTMs with the discovered recurrent neural cells. Also, we set the hidden layer size to 512 and apply variational dropout to the input and output of the RNN layer. We train the network using SGD with a learning rate of 0.1 and a gradient clipping threshold of 5.0. We reduce the learning rate by a factor of 0.25 if the test error does not decrease for 2 epochs.

Table 2 shows a comparison of different methods. Our baseline uses RNN cells generated from random initialized whose F1-score varies greatly and is lower than that of the standard LSTMs. I-DARTS significantly outperforms Random RNNs and DARTS. The best score is achieved when $n = 1$. It indicates that the task prefers a sparse network. Also, we see that our model works with the advanced pre-trained language models in that we replace the LSTM cell to our cell. The I-DARTS architecture yields a new RNN-based state-of-the-art on this task (93.47 F1-score). In Table 2, We find it interesting that Random RNNs are good for NER task. This may result from the design of search space that fit for such tasks substantially. Search space is also a key factor in neural architecture search that new efforts should focus on (Xie et al., 2019a).

We visualize the discovered cells in Figure 3. Each cell is a directed acyclic graph consisting of an ordered sequence of 8 nodes with an activation function applied on each edge. These automatically discovered cells are complex and hard to be designed manually. An interesting phenomenon comes up that the best architecture on language modeling is different from that on name entity recognition. This might result from the fact that different tasks have different inductive bias. Also, this suggests the possibility of architecture selection from the top-k search results on the target task.

4 Related Work

Neural architecture search has been proposed to automatically search for better architectures, showing competitive results on several tasks, e.g., image recognition and language modeling. A s-

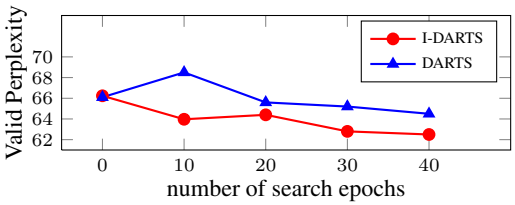


Figure 2: Perplexity vs. search epoch number.

Model	F1
best published	
BiLSTM-CRF (Lample et al., 2016)	90.94
BiLSTM-CRF+ELMo (Peters et al., 2018)	92.22
BERT Base (Devlin et al., 2018)	92.40
BERT Large (Devlin et al., 2018)	92.80
BiLSTM-CRF+PCE (Akbik et al., 2019)	93.18
Random RNNs w/o pre-trained LM	90.64
DARTS w/o pre-trained LM	91.05
I-DARTS ($n = 2$) w/o pre-trained LM	90.96
I-DARTS ($n = 1$) w/o pre-trained LM	91.23
Random RNNs	92.89
DARTS	93.13
I-DARTS ($n = 2$)	93.14
I-DARTS ($n = 1$)	93.47

Table 2: F1 scores on the CoNLL-2003 English NER test set.

trand of NAS research focuses on reinforcement learning (Zoph and Le, 2016) and evolutionary algorithm-based (Xie and Yuille, 2017) methods. They are powerful but inefficient. Recent approaches speed up the search process by weight sharing (Pham et al., 2018) and differentiable architecture search (Liu et al., 2019). But there is no discussion on the softmax-local problem in previous work. Moreover, previous methods are often tested on language modeling. It is rare to see studies on these methods for other NLP tasks.

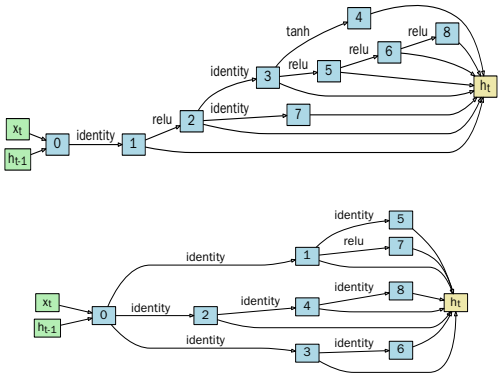


Figure 3: Cells discovered by I-DARTS for language modeling (top) and NER (bottom).

5 Conclusions

We improved the DARTS to address the bias problem by removing the softmax-local constraint. Our method is search efficient and discovers several better architectures for PTB language modeling and CoNLL named entity recognition (NER) tasks. We plan to consider the network density problem in search and apply I-DARTS to more tasks in our future study.

6 Acknowledgments

This work was supported in part by the National Science Foundation of China (Nos. 61876035, 61732005 and 61432013), the National Key R&D Program of China (No. 2019QY1801) and the Opening Project of Beijing Key Laboratory of Internet Culture and Digital Dissemination Research. We also thank the reviewers for their insightful comments.

References

Alan Akbik, Tanja Bergmann, and Roland Vollgraf. 2019. Pooled contextualized embeddings for named entity recognition. In *NAACL*.

Alan Akbik, Duncan Blythe, and Roland Vollgraf. 2018. Contextual string embeddings for sequence labeling. In *COLING*.

Bowen Baker, Otkrist Gupta, Nikhil Naik, and Ramesh Raskar. 2016. Designing neural network architectures using reinforcement learning. *arXiv preprint arXiv:1611.02167*.

Daphne Koller and Nir Friedman. 2009. *Probabilistic Graphical Models - Principles and Techniques*. MIT press.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Zhiheng Huang and Bing Xiang. 2019. Wenet: Weighted networks for recurrent network architecture search. *arXiv preprint arXiv:1904.03819*.

John Lafferty, Andrew McCallum, and Fernando CN Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 260–270, San Diego, California. Association for Computational Linguistics.

Hanxiao Liu, Karen Simonyan, and Yiming Yang. 2019. DARTS: Differentiable architecture search. In *International Conference on Learning Representations*.

Xuezhe Ma and Eduard Hovy. 2016. End-to-end sequence labeling via bi-directional LSTM-CNNs-CRF. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1064–1074, Berlin, Germany. Association for Computational Linguistics.

Stephen Merity, Nitish Shirish Keskar, and Richard Socher. 2018. Regularizing and optimizing LSTM language models. In *International Conference on Learning Representations*.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*.

Matthew Peters, Waleed Ammar, Chandra Bhagavatula, and Russell Power. 2017. Semi-supervised sequence tagging with bidirectional language models. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1756–1765, Vancouver, Canada. Association for Computational Linguistics.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana. Association for Computational Linguistics.

Hieu Pham, Melody Y Guan, Barret Zoph, Quoc V Le, and Jeff Dean. 2018. Efficient neural architecture search via parameter sharing. *arXiv preprint arXiv:1802.03268*.

Erik Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the conll-2003 shared task: Language-independent named entity recognition. In *CoNLL*.

Lingxi Xie and Alan Yuille. 2017. Genetic cnn. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1379–1388.

Saining Xie, Alexander Kirillov, Ross Girshick, and Kaiming He. 2019a. Exploring randomly wired neural networks for image recognition. *arXiv preprint arXiv:1904.01569*.

Sirui Xie, Hehui Zheng, Chunxiao Liu, and Liang Lin. 2019b. SNAS: stochastic neural architecture search. In *International Conference on Learning Representations*.

Zhilin Yang, Zihang Dai, Ruslan Salakhutdinov, and William W. Cohen. 2018. Breaking the softmax bottleneck: A high-rank RNN language model. In *International Conference on Learning Representations*.

Julian G. Zilly, Rupesh Kumar Srivastava, Jan Koutník, and Jürgen Schmidhuber. 2016. Recurrent highway networks. In *ICML*.

Barret Zoph and Quoc V Le. 2016. Neural architecture search with reinforcement learning. *arXiv preprint arXiv:1611.01578*.