Part of Speech Tagging with Neural Architecture Search

Garrett Bingham
Yale University
New Haven, CT
garrett.bingham@yale.edu

Abstract

Recent neural architecture search techniques have achieved state-of-the-art performance in image classification and language modeling, but applications to other machine learning tasks have been lacking. We present the first application of neural architecture search to part of speech tagging. We adapt the DARTS approach to neural architecture search (Liu et al., 2018) to part of speech tagging and achieve encouraging performance. We improve this by introducing BiDARTS, a bidirectional version of DARTS. BiDARTS consistently outperforms DARTS and is competitive with a state-of-the-art part of speech tagger.

1 Introduction

The success of a neural network frequently depends on its architecture. Machine learning experts will often undergo significant trial and error before finally designing an effective neural network. This has inspired recent interest in developing techniques to automatically design effective neural network architectures for a given task. For a recent survey of neural architecture search, see (Elsken et al., 2018b).

Reinforcement learning approaches treat the neural architecture as the agent's action, and an estimate of the architecture's future performance as the agent's reward (Baker et al., 2016; Zoph and Le, 2016; Zoph et al., 2017; Zhong et al., 2018).

Neuro-evolutionary methods, on the other hand, use evolutionary algorithms to optimize the neural architecture, and utilize gradient-based methods to optimize an architecture's weights (Liu et al., 2017; Suganuma et al., 2017; Xie and Yuille, 2017; Real et al., 2017, 2018; Elsken et al., 2018a; Miikkulainen et al., 2019). Evolutionary algorithms evaluate a population of neural network architectures. At each evolution step, architectures are sampled to serve as parents. Mutations are applied to the parent architectures to obtain offspring which are trained and evaluated before being added to the population.

Both reinforcement learning and neuro-evolutionary approaches are computationally expensive. For example, Zoph et al. (2017) required 1800 GPU days of reinforcement learning, while Real et al. (2018) needed 3150 GPU days of evolution. The DARTS approach uses a continuous relaxation of the architecture representation to search for effective architectures using gradient descent in just one GPU day (Liu et al., 2018). We choose DARTS over other neural architecture search approaches because its low computational cost allows us to perform more experiments.

Notable uses of neural architecture search on new tasks include applications to music modeling (Rawal and Miikkulainen, 2018), image restoration (Suganuma et al., 2018), and network compression (Ashok et al., 2017). Others have considered using neural architecture search for multi-task (Liang et al., 2018; Meyerson and Miikkulainen, 2018) and for multi-objective (Elsken et al., 2018a; Dong et al., 2018; Zhou et al., 2018) problems. To the best of our knowledge, neural architecture search has yet to be applied to part of speech tagging. We do so here, and hope that this work will inspire future applications of neural architecture search to important natural language processing tasks.

2 Overview of DARTS

A DARTS recurrent cell is a directed acyclic graph of N ordered nodes, where each node $x^{(i)}$ is a latent representation and each directed edge (i, j) corresponds to an operation $o^{(i,j)}$ that transforms $x^{(i)}$. A cell has two inputs: the input at the current step and the hidden state from the previous step. The output of the cell is obtained by concatenat-

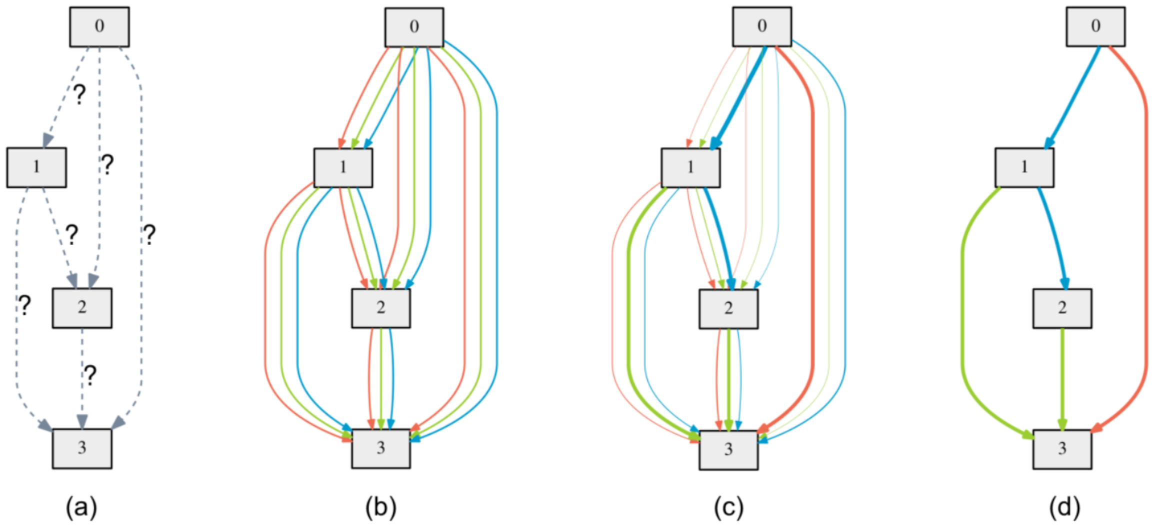

Figure 1: An overview of DARTS: (a) Operations on the edges are initially unknown. (b) Continuous relaxation of the search space by placing a mixture of candidate operations on each edge. (c) Joint optimization of the mixing probabilities and the network weights by solving a bilevel optimization problem. (d) Inducing the final architecture from the learned mixing probabilities. (Liu et al., 2018)

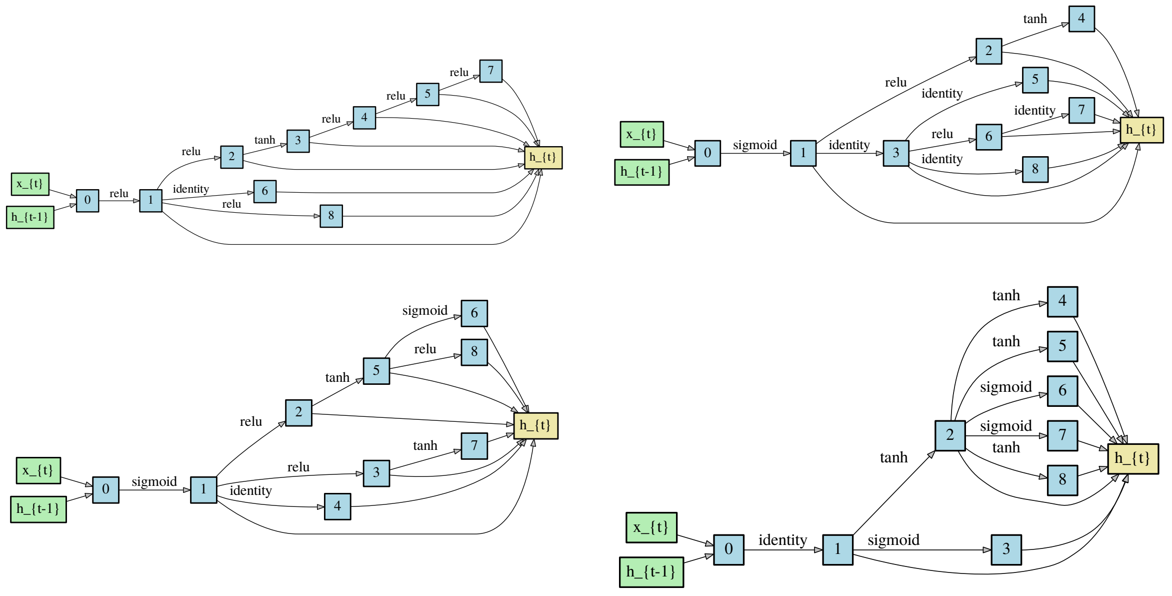

Figure 2: Example DARTS recurrent cells. The left two cells were included in the publicly available DARTS code, while the right two were randomly initialized.

ing all intermediate nodes, where each intermediate node is computed based on the previous nodes.

$$x^{(i)} = \sum_{j<i} o^{(i,j)}(x^{(j)}) \tag{1}$$

Let $\mathcal{O}$ be the set of all operations. For DARTS recurrent cells, $\mathcal{O} = \{$zero, tanh, relu, sigmoid, identity$\}$, where the *zero* operation indicates the absence of an edge between two nodes. To make the search space continuous, the choice of a particular operation is represented as a softmax over all possible operations.

$$\bar{o}^{(i,j)}(x) = \sum_{o \in \mathcal{O}} \frac{\exp(\alpha_o^{(i,j)})}{\sum_{o' \in \mathcal{O}} \exp(\alpha_{o'}^{(i,j)})} o(x) \tag{2}$$

The mixture of operations for an edge (i, j) is parameterized by the vector $\alpha^{(i,j)}$ of dimension $|\mathcal{O}|$. The task of architecture search becomes a task in learning the set of continuous variables $\alpha = \{\alpha^{(i,j)}\}$. Discrete architectures are obtained by replacing each mixed operation $\bar{o}^{(i,j)}$ with the most likely operation $o^{(i,j)} = \text{argmax}_{o \in \mathcal{O}} \alpha_o^{(i,j)}$.

Let $\mathcal{L}_{train}$ be the training loss and $\mathcal{L}_{val}$ be the validation loss. DARTS attempts to find the architecture α^* that minimizes the validation loss $\mathcal{L}_{val}(w^*, \alpha^*)$, where the weights w^* associated with the architecture minimize the training loss $w^* = \text{argmin}_w \mathcal{L}_{train}(w, \alpha^*)$. This corresponds to the bilevel optimization problem:

$$\min_{\alpha} \quad \mathcal{L}_{val}(w^*(\alpha), \alpha) \tag{3}$$

$$\text{s.t.} \quad w^*(\alpha) = \text{argmin}_w \mathcal{L}_{train}(w, \alpha) \tag{4}$$

It is expensive to solve the bilevel optimization problem exactly, so DARTS compromises by alternating gradient steps in the weights w and in the architecture representation α. The weights are optimized by descending in the direction $\nabla_w \mathcal{L}_{train}(w, \alpha)$, while the architecture is optimized by descending in the direction $\nabla_\alpha \mathcal{L}_{val}(w - \xi \nabla_w \mathcal{L}_{train}(w, \alpha), \alpha)$, where ξ is set equal to the learning rate for the weights optimizer. This algorithm does not necessarily converge, and as a result DARTS is sensitive to the initial random seed.

The process of searching for architectures is illustrated in Figure 1. After the architecture search phase is complete, the best performing architecture is retrained from scratch. While DARTS is capable of creating both convolutional and recurrent cells, we focus on recurrent cells in this paper. Example DARTS recurrent cells are shown in Figure 2. For more details, see the DARTS paper (Liu et al., 2018).

3 Experiments

We compare DARTS tagging accuracy with that of jPTDP (Nguyen and Verspoor, 2018), which outperformed the state of the art on the benchmark English Penn treebank (Prasad et al., 2008). jPTDP uses a two-layer bidirectional LSTM followed by a MLP and a softmax layer to predict part of speech tags. For fairness, we parse the `.conllu` input files with code from jPTDP[1] and feed this as input to DARTS.

We also introduce BiDARTS, a bidirectional version of DARTS. To the best of our knowledge, this is the first neural architecture search approach that searches for bidirectional recurrent models. In BiDARTS, we train a forward and reverse DARTS model separately, and sum the outputs to get the final prediction. It is possible to consider one architecture for the forward model and a different architecture for the reverse model, but we leave this for future work

We apply DARTS, BiDARTS, and jPTDP to part of speech tagging on Universal Dependencies v2.2 treebanks.[2] Among the results listed in the jPTDP paper (Nguyen and Verspoor, 2018), we consider the 60 treebanks for which there exist a train, dev, and test `.conllu` file. For DARTS and BiDARTS, we report the best accuracy among the four architectures depicted in Figure 2. It is possible that higher accuracy could be obtained by searching for additional architectures or by considering DARTS cells with a different number of intermediate nodes, but because of computational cost, we do not do so here.

We train DARTS and BiDARTS for 50 epochs, and leave all other parameters to their default values. This includes a batch size of 64 with learning rate $\eta_w = 20$ and weight decay 8×10^{-7}. Embedding and hidden sizes are set to 850, and BPTT length is 35. DARTS also uses variational dropout of 0.2 on word embeddings, 0.75 on the cell input, 0.25 on the hidden nodes, and 0.75 on the output layer. These values were chosen for comparison with previous neural architecture search papers. The results from jPTDP are taken directly from

[1] `https://github.com/datquocnguyen/jPTDP`

[2] `http://universaldependencies.org`

Treebank	DARTS	BiDARTS	jPTDP-c	jPTDP
Afrikaans-AfriBooms	91.95	93.54	94.58	95.73
Ancient_Greek-Perseus	82.59	83.20	82.39	88.95
Ancient_Greek-PROIEL	92.42	92.60	92.83	96.05
Arabic-PADT	93.59	93.49	94.85	96.33
Basque-BDT	90.10	90.25	89.57	93.62
Bulgarian-BTB	93.38	95.55	95.63	98.07
Catalan-AnCora	93.51	94.94	96.30	98.46
Chinese-GSD	88.00	89.26	91.18	93.26
Croatian-SET	90.55	92.34	93.25	97.42
Czech-CAC	93.61	94.35	95.31	98.87
Czech-FicTree	92.83	94.47	94.87	97.98
Czech-PDT	95.59	96.46	97.26	98.74
Danish-DDT	90.02	91.27	93.06	96.18
Dutch-Alpino	88.98	90.58	92.59	95.62
Dutch-LassySmall	89.66	91.20	91.41	95.21
English-EWT	90.56	92.47	93.69	95.48
English-GUM	86.23	87.79	88.67	94.10
English-LinES	89.98	92.33	92.54	95.55
Estonian-EDT	89.68	90.31	91.51	96.87
Finnish-FTB	83.54	87.03	86.25	94.53
Finnish-TDT	88.93	89.43	89.24	96.12
French-GSD	93.84	94.87	96.08	97.11
French-Sequoia	93.90	94.40	95.94	97.92
French-Spoken	89.09	89.35	90.14	94.25
Galician-CTG	93.62	94.82	95.45	97.12
German-GSD	87.60	90.10	91.47	94.07
Gothic-PROIEL	92.05	92.89	91.98	93.45
Greek-GDT	90.18	91.41	93.34	96.59
Hebrew-HTB	92.41	93.63	94.31	96.24
Hindi-HDTB	94.45	95.55	95.97	96.94
Hungarian-Szeged	78.57	79.15	79.19	92.07
Indonesian-GSD	88.50	88.92	89.13	93.29
Italian-ISDT	94.17	95.48	96.41	98.01
Italian-PoSTWITA	90.46	91.86	92.15	95.41
Japanese-GSD	92.92	94.52	95.15	97.27
Korean-GSD	83.16	84.91	83.93	93.35
Korean-Kaist	81.40	84.22	83.27	93.53
Latin-ITTB	95.64	96.86	97.24	98.12
Latin-PROIEL	91.79	92.47	92.01	95.54
Latvian-LVTB	85.67	87.46	86.49	93.53
Norwegian-Bokmaal	90.85	93.21	95.07	97.73
Norwegian-Nynorsk	90.42	93.00	94.99	97.33
Old_Church_Slavonic-PROIEL	91.54	91.85	90.71	93.69
Old_French-SRCMF	90.85	93.25	93.59	95.12
Persian-Seraji	94.79	95.32	95.48	96.66
Polish-LFG	90.39	91.74	92.67	98.22
Polish-SZ	86.58	88.70	89.57	97.05
Portuguese-Bosque	91.10	93.39	94.16	96.76
Romanian-RRT	93.48	94.46	95.00	97.43
Russian-SynTagRus	93.54	94.89	95.77	98.51
Serbian-SET	90.24	91.84	93.21	97.40
Slovak-SNK	80.37	81.94	83.11	95.18
Slovenian-SSJ	88.30	91.05	92.60	97.79
Spanish-AnCora	94.33	95.10	96.30	98.57
Swedish-Talbanken	90.10	92.99	93.81	96.55
Turkish-IMST	87.05	88.49	86.94	92.93
Ukrainian-IU	83.23	84.57	85.63	95.24
Urdu-UDTB	90.91	92.36	92.72	93.35
Uyghur-UDT	83.78	87.52	84.93	87.63
Vietnamese-VTB	88.12	88.04	86.68	87.63
AVERAGE	**89.92**	**91.32**	**91.83**	**95.63**

Table 1: Part of speech tagging accuracy for DARTS (Liu et al., 2018), BiDARTS, and the state-of-the-art tagger jPTDP (Nguyen and Verspoor, 2018) on Universal Dependencies v2.2 treebanks. jPTDP-c corresponds to jPTDP without character embeddings.

the paper (Nguyen and Verspoor, 2018).

jPTDP takes both word and character embeddings as input. Since DARTS only accepts word tokens as input, we also evaluate the performance of jPTDP without character embeddings. This provides a more fair comparison with DARTS. To do so, we downloaded the publicly available code for jPTDP, removed the character embeddings, and trained a model on each treebank leaving all parameters to their default values. We leave modifying DARTS to accept character embeddings to future work.

4 Discussion

The results are summarized in Table 1. Notice that BiDARTS outperforms DARTS in almost every case. As the first bidirectional neural architecture search approach, this is an important result. It suggests that future neural architecture search techniques should consider bidirectional recurrent cells over unidirectional ones whenever feasible.

Although DARTS and BiDARTS are unable to outperform jPTDP with character embeddings, they still achieve satisfactory performance. Additionally, the performance should be contrasted with the amount of human effort required to design the neural network architectures. While jPTDP required significant time and human effort, the DARTS and BiDARTS architectures were discovered in one GPU day by executing a single command. This demonstrates that neural architecture search approaches can be useful, especially when the cost of human labor far outweighs the need for state-of-the-art performance.

Notably, BiDARTS achieves comparable accuracy with jPTDP when character embeddings are removed. This suggests that the presence of the character embeddings was a principal factor in jPTDP obtaining higher accuracy, and that BiDARTS is effective at part of speech tagging.

5 Conclusion

We present the first application of neural architecture search to part of speech tagging and introduce BiDARTS, the first bidirectional neural architecture search approach. BiDARTS is competitive with a state-of-the-art part of speech tagger. Future work will include applying BiDARTS to new machine learning tasks.

References

Anubhav Ashok, Nicholas Rhinehart, Fares Beainy, and Kris M Kitani. 2017. N2n learning: network to network compression via policy gradient reinforcement learning. *arXiv preprint arXiv:1709.06030*.

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

Jin-Dong Dong, An-Chieh Cheng, Da-Cheng Juan, Wei Wei, and Min Sun. 2018. Dpp-net: Device-aware progressive search for pareto-optimal neural architectures. *arXiv preprint arXiv:1806.08198*.

Thomas Elsken, Jan Hendrik Metzen, and Frank Hutter. 2018a. Efficient multi-objective neural architecture search via lamarckian evolution. *arXiv preprint arXiv:1804.09081*.

Thomas Elsken, Jan Hendrik Metzen, and Frank Hutter. 2018b. Neural architecture search: A survey. *arXiv preprint arXiv:1808.05377*.

Jason Liang, Elliot Meyerson, and Risto Miikkulainen. 2018. Evolutionary architecture search for deep multitask networks. *arXiv preprint arXiv:1803.03745*.

Hanxiao Liu, Karen Simonyan, Oriol Vinyals, Chrisantha Fernando, and Koray Kavukcuoglu. 2017. Hierarchical representations for efficient architecture search. *arXiv preprint arXiv:1711.00436*.

Hanxiao Liu, Karen Simonyan, and Yiming Yang. 2018. Darts: Differentiable architecture search. *arXiv preprint arXiv:1806.09055*.

Elliot Meyerson and Risto Miikkulainen. 2018. Pseudo-task augmentation: From deep multitask learning to intratask sharing—and back. *arXiv preprint arXiv:1803.04062*.

Risto Miikkulainen, Jason Liang, Elliot Meyerson, Aditya Rawal, Daniel Fink, Olivier Francon, Bala Raju, Hormoz Shahrzad, Arshak Navruzyan, Nigel Duffy, et al. 2019. Evolving deep neural networks. In *Artificial Intelligence in the Age of Neural Networks and Brain Computing*, pages 293–312. Elsevier.

Dat Quoc Nguyen and Karin Verspoor. 2018. An improved neural network model for joint pos tagging and dependency parsing. *arXiv preprint arXiv:1807.03955*.

Rashmi Prasad, Nikhil Dinesh, Alan Lee, Eleni Miltsakaki, Livio Robaldo, Aravind K Joshi, and Bonnie L Webber. 2008. The penn discourse treebank 2.0. In *LREC*. Citeseer.

Aditya Rawal and Risto Miikkulainen. 2018. From nodes to networks: Evolving recurrent neural networks. *arXiv preprint arXiv:1803.04439*.

Esteban Real, Alok Aggarwal, Yanping Huang, and Quoc V Le. 2018. Regularized evolution for image classifier architecture search. *arXiv preprint arXiv:1802.01548*.

Esteban Real, Sherry Moore, Andrew Selle, Saurabh Saxena, Yutaka Leon Suematsu, Jie Tan, Quoc V Le, and Alexey Kurakin. 2017. Large-scale evolution of image classifiers. In *International Conference on Machine Learning*, pages 2902–2911.

Masanori Suganuma, Mete Ozay, and Takayuki Okatani. 2018. Exploiting the potential of standard convolutional autoencoders for image restoration by evolutionary search. *arXiv preprint arXiv:1803.00370*.

Masanori Suganuma, Shinichi Shirakawa, and Tomoharu Nagao. 2017. A genetic programming approach to designing convolutional neural network architectures. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 497–504. ACM.

Lingxi Xie and Alan Yuille. 2017. Genetic cnn. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 1388–1397. IEEE.

Zhao Zhong, Junjie Yan, Wei Wu, Jing Shao, and Cheng-Lin Liu. 2018. Practical block-wise neural network architecture generation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2423–2432.

Yanqi Zhou, Siavash Ebrahimi, Sercan Ö Arık, Haonan Yu, Hairong Liu, and Greg Diamos. 2018. Resource-efficient neural architect. *arXiv preprint arXiv:1806.07912*.

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

Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V Le. 2017. Learning transferable architectures for scalable image recognition. *arXiv preprint arXiv:1707.07012*, 2(6).