Master's Thesis of Engineering

Differentiable Modular Learning For Multi-Task Learning

다중 작업 학습을 위한 미분 가능한 모듈 학습법

August 2022

Graduate School of Engineering
Seoul National University
Computer Science and Engineering

Joonho Yeom

Differentiable Modular Learning For Multi-Task Learning

다중 작업 학습을 위한 미분 가능한 모듈 학습법

지도교수 문 병 로

이 논문을 공학석사 학위논문으로 제출함

2022 년 6 월

서울대학교 대학원

컴퓨터공학부

염 준 호

염 준 호 의 공학석사 학위논문을 인준함

2022 년 7 월

위 원 장	이 광 근
부위원장	문 병 로
위 원	허 충 길

Abstract

While recent advances in deep learning enable to solve complex problems such as computer vision, natural language processing, and scientific discovery, they still depend on the huge amount of computations for high quality results. As a way of enlarging the capacity of a neural architecture, one can train a single supernet for multiple tasks to exploit the shareable knowledge among the tasks. In the recent works on multi-task learning, sparse sharing has drawn attention for its parameter efficiency with less negative transfer effect compared to the traditional sharing strategies. However, existing methods of sparse sharing have limitations in that finding a modularization scheme for the specific task is extremely computational heavy. In this work, we alleviate the cost of searching the modularization schemes by introducing differentiable modular learning which finds the schemes for all tasks in one-shot through continuous relaxation. We demonstated on three sequence labeling tasks on CoNLL-2003 dataset and show our approach achieves the average accuracy on par with the accuracy of the previous work whereas finding the modularization schemes much faster. Lastly, we analyze the performance degradation according to the sparsity level of the schemes and study the overlapping ratio between schemes.

Keywords: Multi-Task Learning, Modular Learning, Neural Architecture Search
Student Number: 2020-23215

Contents

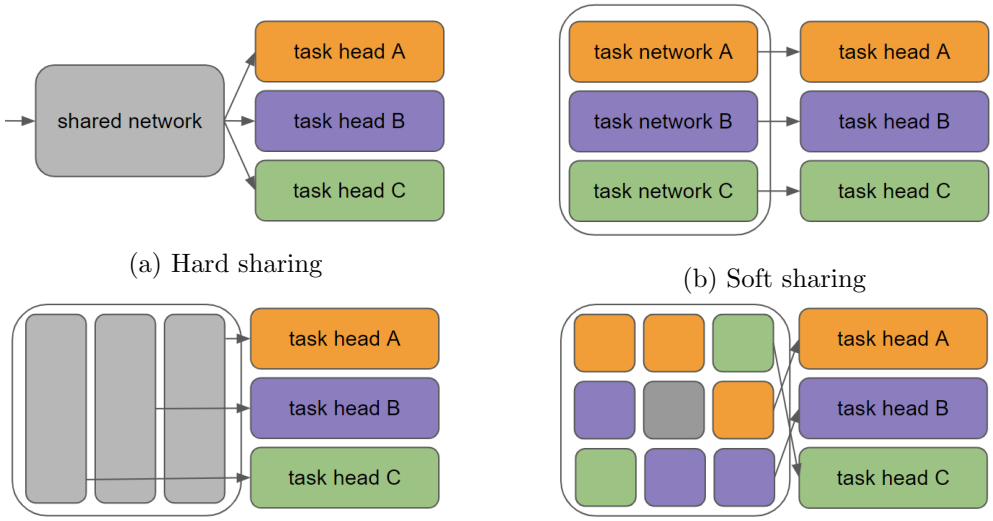
List of Figures

List of Tables

Chapter 1

Introduction

Traditional supervised learning approach has shown remarkable success in a variety of domains such as computer vision [2, 3] and natural language processing [4, 5]. Along with this success, the size of neural network and the required computation cost also exponentially increased. To tackle this problem, diverse approaches have been proposed. For a constructive approach, *EfficientNet* [6] devised the efficient scale-up strategy for convolutional neural networks called *compound scaling*. *Compound scaling* effectively scale up the network with the golden ratio of width, height and depth. As a destructive approaches, network quantization [7, 8, 9] and pruning [10, 11] have attracted a large amount of attention. They increased parameter efficienty of neural network by cutting down the less important part of the neural network. On top of that, beyond the handcrafted effort, there were attemps to find the novel efficient architectures rather than manipulating existing architectures. Multi-Objective Neural Architecture Search [12, 13] automatically designed the efficient neural architecture under the constraints including the number of paramters, inference latency, accuracy.



(a) Hard sharing

(b) Soft sharing

(c) Hierarchical sharing

(d) Sparse sharing

Figure 1.1: Sharing Mechanisms of Multi-Task Learning

One of the endeavors to extend the efficiency of neural network is multi-task learning. The basic motivation of multi-task learning is to leverage the shareable knowledge acquired by training given tasks simultaneously. The parameter sharing mechanisms of multi-task learning can be classified into four categories: (a) *hard sharing* approaches [14, 15] have task specific heads on the shared feature extractor. (b) *soft sharing* approaches [16, 17] allows each tasks to have separate model, but they can access the parameters of others. (c) *hierarchical sharing* approaches [18, 19] put task specific heads at different layers. (d) *sparse sharing* approaches [1] have similar topologies with hard sharing but allow each task to use the partial components of the feature extractor. Even though the sparse sharing has proven its architectural expressivity and parameter efficiency [1], existing methods of this approach still have critical flaw that finding the modularization scheme for each task is too computationally expensive as the number of tasks grows.

In this work, we propose *Differentiable Modular Learning* approach which finds the modularization schemes for all tasks in one-shot with near equivalent performance of the previous work [1] and demonstrate its sparsity, corresponding performance and the overlapping ratio among the modularization schemes of given tasks.

Chapter 2

Related Work

For sparse sharing of a supernet, Devin et al. [20] utilized structured networks but they trained network with the predefined fixed composition schemes, one related to the robot and the other to the task. Alet et al. [21] jointly optimized the modular network and the composition scheme of modules. They searched the modularization schemes for the tasks using local search and Monte-Carlo sampling. Local search is reasonable choice regarding the fact that the neural architecture has locality property [22, 23, 24]. However, this approach cannot be accepted when the search space of the possible modularization schemes is non-trivial. For example, the search space size of the weight-level modularization is exponential to the number of parameters of neural network.

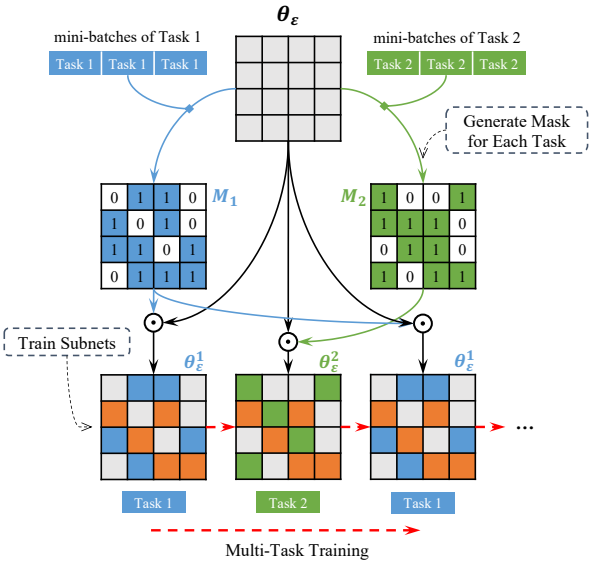


Figure 2.1: Illustration of Sun et al.,[1] approach to learn sparse sharing architectures.

Sun et al. [1] also jointly searched the modularization scheme and the shared weight in multi-task learning setup. Their search strategy is inspired by the *lottery ticket hypothesis* [10] which states that the over parameterized network contains the compact sub-network whose accuracy is on par with that of its supernet. They find the modularization scheme for each task using the *Iterative Magniture Pruning* [10] algorithm. The IMP algorithm iteratively trains the network from the initial weight to find the which neuron has large magnitude in the pruned architecture. This approach promises high accuracy sparse sub-network but consumes heavy computations, since they requires iterative training of supernet per task to find a scheme for the task. Moreover, after finding the schemes, they need to run a multi-task training from scratch with the found schemes to obtain the shareable weight.

Chapter 3

Method

In this section, we present *Differentiable Modular Learning* method which jointly optimizes the model weights and the modularization scheme parameters in one-shot manner.

3.1 Problem Setup

Multi-task learning(MTL) retreives shareable information by training all tasks simulatneously. A typical formulation of the loss function of MTL can be written as

$$\mathcal{L}(\boldsymbol{\theta}) = \sum_{\tau} \alpha_{\tau} \mathcal{L}_{\tau}(f_{\boldsymbol{\theta}}, D_{\tau}) \tag{3.1}$$

where $f_{\boldsymbol{\theta}}$ is the $\boldsymbol{\theta}$-parameterized supernet, τ denotes a task, D_{τ} is the dataset for the task τ, and α_{τ} is a hyperparameter denoting the proportion of the task among all target tasks. Similarly, we can extend this formular to the sparse sharing mechanism by applying the modularization scheme as Equation 3.2.

In our work, we perform weight-level modularization, then the modularization scheme can be represented by binary matrix $\mathbf{M}_\tau \in \{0,1\}^{|\boldsymbol{\theta}|}$. Each element of the scheme matrix determines wheter the neuron to be skipped or not.

$$\mathcal{L}(\mathbf{M}_1, \ldots, \mathbf{M}_T, \boldsymbol{\theta}) = \sum_{\tau=1}^{T} \alpha_\tau \mathcal{L}_\tau(f_{\boldsymbol{\theta} \odot \mathbf{M}_\tau}, D_\tau) \tag{3.2}$$

where $\odot$ is the element-wise multiplication. Then the optimization problem can be formulated as

$$\min_{\mathbf{M}_1, \ldots, \mathbf{M}_T, \boldsymbol{\theta}} \sum_{\tau=1}^{T} \alpha_\tau \mathcal{L}_\tau(f_{\boldsymbol{\theta} \odot \mathbf{M}_\tau}, D_\tau) \tag{3.3}$$

3.2 Differentiable Modular Learning

Solving directly discrete-continuous joint optimization problem for multiple tasks (Equation 3.3) is prohibitively expensive. In weight-level scheme search, each modularization scheme matrices has $2^{|\boldsymbol{\theta}|}$ candidates. To ease the computational budget, instead of searching $\mathbf{M}_\tau$ in discrete search space, we utilized gradient guided search by applying the continuous relaxation. The Equation 3.4 describes the coninuous relaxed optimization problem.

$$\min_{\boldsymbol{\pi}_1, \ldots, \boldsymbol{\pi}_T, \boldsymbol{\theta}} \sum_{\tau=1}^{T} \alpha_\tau \mathbb{E}_{\mathbf{M}_\tau \sim p_{\boldsymbol{\pi}_\tau}} \mathcal{L}_\tau(f_{\boldsymbol{\theta} \odot \mathbf{M}_\tau}, D_\tau) \tag{3.4}$$

where $\mathbf{M}_\tau$ is a random variable sampled from distribution $p_{\boldsymbol{\pi}_\tau}$ and $p_{\boldsymbol{\pi}_\tau}$ is Bernoulli distribution parameterized by $\boldsymbol{\pi}_\tau \in [0,1]^{|\boldsymbol{\theta}|}$. From this relaxation, we can jointly optimize the supernet parameter $\boldsymbol{\theta}$ and modularization scheme distribution parameters $\boldsymbol{\pi}_\tau$. However, since the derivetives of the binary mask are almost zero, we cannot take the meaningful gradient for updating the probability mask. To overcome this hindrance, we borrow the approximation trick-*Straight-Through Estimator*(STE) [25] popularly used in network quantization and pruning literature [26, 27]. The STE trick replaces zero gradients of a

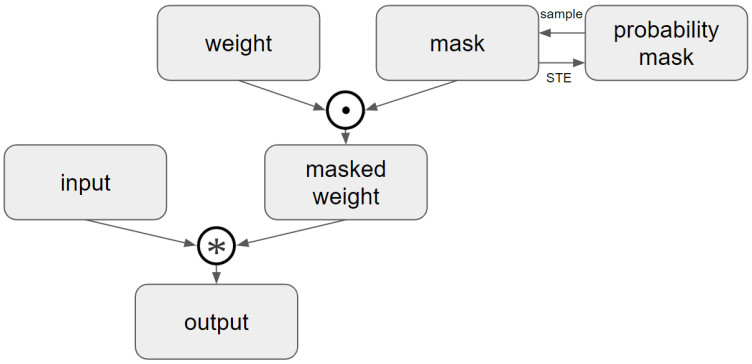


Figure 3.1: STE applied computational graph

discrete mask with the incomming gradient as if the mask operation was an identity function (Fig 3.1).

Under this framework, we jointly train $\boldsymbol{\theta}, \boldsymbol{\pi}_\tau$ using SGD with learning rate $\eta, \eta_{\boldsymbol{\pi}}$, respectively. For training the relaxed scheme $\boldsymbol{\pi}$, we sample the discrete scheme $\mathbf{M}_\tau$ from $\boldsymbol{\pi}_\tau$ for every mini-batch and compute forward pass and perform backpropagation utilizing the STE trick. For training the $\boldsymbol{\theta}$, each tasks only updates the parameters activated by its own scheme. The entire process is summarized in Algorithm 1

Algorithm 1 Differentiable Modular Learning

Randomly initialize $p_{\boldsymbol{\pi}_1}, \ldots, p_{\boldsymbol{\pi}_\tau}, \boldsymbol{\theta}$,
for iter $= 1, \cdots,$ N **do**
  Sample a batch of data $\mathbf{x}$
  for task $\tau = 1, \cdots, T$ **do**
    Sample a mask $\mathbf{M}_\tau \sim p_{\boldsymbol{\pi}_\tau}$ for the task τ
    Forward pass $f(\mathbf{x}; \boldsymbol{\theta} \odot \mathbf{M}_\tau)$
    Backward pass to calculate $\frac{\partial \mathcal{L}}{\partial \boldsymbol{\theta}}, \frac{\partial \mathcal{L}}{\partial \mathbf{M}_\tau}$
    $\Delta_{\boldsymbol{\theta}} = \Delta_{\boldsymbol{\theta}} + \frac{\partial \mathcal{L}_\tau}{\partial \boldsymbol{\theta}}$, $\Delta_{\boldsymbol{\pi}_\tau} \overset{\text{STE}}{\leftarrow} \frac{\partial \mathcal{L}_\tau}{\partial \mathbf{M}_\tau}$
  end for
  $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \eta \Delta_{\boldsymbol{\theta}}$
  $\boldsymbol{\pi}_\tau \leftarrow \boldsymbol{\pi}_\tau - \eta_\pi \Delta_{\boldsymbol{\pi}_\tau}$ for $\tau = 1, \cdots, T$
end for
return Trained $\boldsymbol{\theta}$ and discretized mask from the schemes $\boldsymbol{\pi}_{1,\ldots,T}$ according to the target sparsity

Chapter 4

Experiments

4.1 Experiment Setup

We presents results for *Differentiable Modular learning* on three sequence labeling tasks for CoNLL-2003 [28]: Part-Of-Speech (POS), Named Entity Recognition (NER) and Chunking. CoNLL-2003 dataset consists of 204,576, 51,578, 46,666 tokens for the train, validate, and test respectively. For fair comparison with the previous work [1], we synchronize the hyperparameters and the baseline model with it. Baseline model is a CNN-BiLSTM [29] with the hyperparameters described in (Table 4.1). We used SGD optimizer for the model weight training and employed Adam [30] optimizer for the scheme parameter training.

Hyper-parameters	
Embedding dimension	100
Convolution width	3
CNN output size	30
LSTM hidden size	200
SGD Learning rate	0.1
Dropout	0.5
Mini-batch size	10
Adam Learning rate	3e-3
Adam Weight decay	1e-5

Table 4.1: Hyper-parameters used in our experiments

4.2 Main Results

Systems	POS		NER		Chunking		# Params
	Test Acc.	Δ	Test F1	Δ	Test F1	Δ	
Single	95.09	-	89.36	-	89.92	-	1602k
Single (subnet)	95.11	+0.02	89.39	+0.03	89.96	+0.04	811k
Hard	95.34	+0.25	88.68	−0.68	90.92	+1.00	534k
Soft	95.16	+0.07	89.35	−0.01	90.71	+0.79	1596k
Hierarchical	95.09	+0.00	89.30	−0.06	90.89	+0.97	1497k
Sun et al.[1]	95.56	+0.47	90.35	+0.99	91.55	+1.63	396k
Ours	95.46	+0.37	89.56	+0.2	91.52	+1.6	396k

Table 4.2: Comparison with different sharing mechanisms and the previous work [1] on CoNLL-2003

To compare the performance of the strategies on the same line, we discretized the modularization scheme with the same level sparsity as the Sun et al. work. In *Differentiable Modular Learning*, since the scheme mask is a real-valued matrix, we can easily control the sparsity level with the order of probability. In our experiment, we could reach the near performance of the Sun et al. [1] work, while Sun et al., method took 72 GPU hours to find all the three schemes, whereas our method only took 2 GPU hours.
On top of that, we carried out a experiment about the performance degradation over sparsity degree.

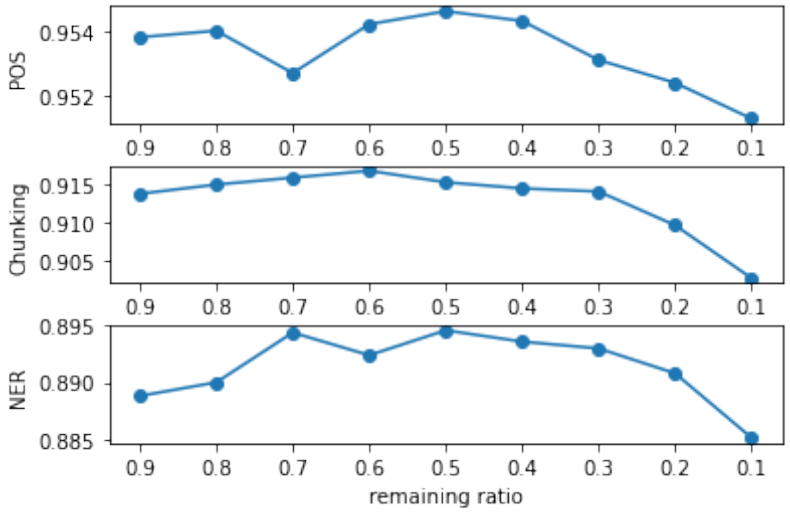


Figure 4.1: Performance corresponding to the sparsity level

The results shows that the each tasks suffers from the performance degradation as the remaining ratio diminished under 50% and the slight performance losses were also seen in the high ratio region above 50%. This suggests that highly overlapped schemes are susceptible to the negative transfer effect whereas the sparsely shared schemes more effectively expoited the supernet parameters.

	POS	NER	Chunking
Sun et al. [1]			
From scratch	95.36	89.62	91.04
Warm-up	95.56	90.35	91.55
Ours			
From scratch	95.41	89.44	91.02
Warm-up	95.46	89.56	91.52

Table 4.3: Comparison between Warm-up and From scratch training on test accuracy (for POS) and F1 (for NER and Chunking) on CoNLL-2003

As many research pointed out, evaluating the true qualities of the architectures demands the adequatly trained weights [22, 31, 24]. Inspired by this perspectives, we could guess the warm-up would catalyze the scheme search process. To figure it out qualitatively, we conducted an ablation study. In from scractch setting, we started scheme search process by randomly initialized weights and in warm-up setting, we initialized the weights of the supernet trained for 10 epochs for a single task. The result confirmed that the warm-up achieves consistently better accuracy for all tasks.

Chapter 5

Conculsion and Discussion

Sparse sharing boosts the parameter efficiency of multi-task learning by modularizing the components of the supernet. Allowing the unrelated tasks not to share all the parameters, it avoids negative transfer effect and each task can utilize only a portion of parameters of the supernet. However, since the methods proposed by previous works have heavy computation issues on finding the modularization schemes for each tasks, it was not applicable for many tasks with a large scheme search space cases. To ease this burden, we introduce *Differentiable Modular Learning* which relaxes the discrete scheme search space to the continuous search space. Owing to the gradient guided search on continuous space, we could achieve the near equal performance as the previous work much faster time with the same sparsity level for the three sequence labeling tasks.
Even though we can take advantage of gradient-based search, the cost of scheme search still grows linearly to the number of tasks. One of the main purposes of multi-task learning might be of obtaining a general network. Therefore, the fast adaptation to unseen tasks is a desirable property. In perspective of model

parameters, we can quickly adapt to the unseen examples with the pre-trained model. Similary, if one can know the task similarity without training, we could also effectively reduce the scheme search cost by exploiting the existing schemes as a good initialization for scheme search.

Moreover, as Alet et al. [21] shows the effectiveness of local search for scheme search, evolutionary operation can be another promising method in this field. Since the continuous relaxation is approximation, there might be the gap between the true optimality and the optimality in the relaxed space. It is worthwhile to pay more attention to hybrid of evolutionary approach and local search.

Appendix A

How accurately does STE approximate?

Straight-Through Estimator helps us to retrieve the gradient signal from the zero gradient of discrete mask. However this is not a true gradient but a linear approximation. To investigate how closely can we reach to the true distribution through STE approximation, we conduct a simple yet informative experiment. From the uniform randomly $\sim Unif(0, 1)$ initialized 1,000 dimensional vector, we generated the discrete mask dataset through Bernoulli sampling. With these data, we tried to fit the original random vector using Adam with STE for 500 epochs.

difference > 0.05	difference > 0.01	average difference
0%	$20.26 \pm 2.13\%$	0.0063 ± 0.0001

Table A.1: STE experiment

The table shows the difference between true underlying distribution and the

fitted distribution. We repeated the experiment three times and reported the mean with the deviations. Adam with STE nearly fitted all elements in the random vector with the small error bound. This suggest that even though STE cannot reach the exact true distribution, but it provides an exellent approximation for finding the binary mask.

Bibliography

[1] T. Sun, Y. Shao, X. Li, P. Liu, H. Yan, X. Qiu, and X. Huang, "Learning sparse sharing architectures for multiple tasks," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, pp. 8936–8943, Apr. 2020.

[2] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, "An image is worth 16x16 words: Transformers for image recognition at scale," in *International Conference on Learning Representations*, 2021.

[3] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo, "Swin transformer: Hierarchical vision transformer using shifted windows," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021.

[4] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei, "Language models are few-shot learners," in *Advances in Neural Information Processing Systems* (H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, eds.), vol. 33, pp. 1877–1901, Curran Associates, Inc., 2020.

[5] A. Chowdhery, S. Narang, J. Devlin, M. Bosma, G. Mishra, A. Roberts, P. Barham, H. W. Chung, C. Sutton, S. Gehrmann, P. Schuh, K. Shi, S. Tsvyashchenko,

J. Maynez, A. Rao, P. Barnes, Y. Tay, N. Shazeer, V. Prabhakaran, E. Reif, N. Du, B. Hutchinson, R. Pope, J. Bradbury, J. Austin, M. Isard, G. Gur-Ari, P. Yin, T. Duke, A. Levskaya, S. Ghemawat, S. Dev, H. Michalewski, X. Garcia, V. Misra, K. Robinson, L. Fedus, D. Zhou, D. Ippolito, D. Luan, H. Lim, B. Zoph, A. Spiridonov, R. Sepassi, D. Dohan, S. Agrawal, M. Omernick, A. M. Dai, T. S. Pillai, M. Pellat, A. Lewkowycz, E. Moreira, R. Child, O. Polozov, K. Lee, Z. Zhou, X. Wang, B. Saeta, M. Diaz, O. Firat, M. Catasta, J. Wei, K. Meier-Hellstern, D. Eck, J. Dean, S. Petrov, and N. Fiedel, "Palm: Scaling language modeling with pathways," 2022.

[6] M. Tan and Q. Le, "EfficientNet: Rethinking model scaling for convolutional neural networks," in *Proceedings of the 36th International Conference on Machine Learning* (K. Chaudhuri and R. Salakhutdinov, eds.), vol. 97 of *Proceedings of Machine Learning Research*, pp. 6105–6114, PMLR, 09–15 Jun 2019.

[7] B. Jacob, S. Kligys, B. Chen, M. Zhu, M. Tang, A. Howard, H. Adam, and D. Kalenichenko, "Quantization and training of neural networks for efficient integer-arithmetic-only inference," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[8] Y. Li, R. Gong, X. Tan, Y. Yang, P. Hu, Q. Zhang, F. Yu, W. Wang, and S. Gu, "{BRECQ}: Pushing the limit of post-training quantization by block reconstruction," in *International Conference on Learning Representations*, 2021.

[9] C. Guo, Y. Qiu, J. Leng, X. Gao, C. Zhang, Y. Liu, F. Yang, Y. Zhu, and M. Guo, "SQuant: On-the-fly data-free quantization via diagonal hessian approximation," in *International Conference on Learning Representations*, 2022.

[10] J. Frankle and M. Carbin, "The lottery ticket hypothesis: Finding sparse, trainable neural networks," in *ICLR*, 2019.

[11] F. Meng, H. Cheng, K. Li, H. Luo, X. Guo, G. Lu, and X. Sun, "Pruning filter in filter," in *Advances in Neural Information Processing Systems* (H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, eds.), vol. 33, pp. 17629–17640, Curran Associates, Inc., 2020.

[12] B. Wu, X. Dai, P. Zhang, Y. Wang, F. Sun, Y. Wu, Y. Tian, P. Vajda, Y. Jia, and K. Keutzer, "Fbnet: Hardware-aware efficient convnet design via differentiable neural architecture search," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[13] T. Elsken, J. H. Metzen, and F. Hutter, "Efficient multi-objective neural architecture search via lamarckian evolution," in *International Conference on Learning Representations*, 2019.

[14] R. Collobert and J. Weston, "A unified architecture for natural language processing: deep neural networks with multitask learning," in *ICML*, pp. 160–167, 2008.

[15] X. Liu, P. He, W. Chen, and J. Gao, "Multi-task deep neural networks for natural language understanding," in *ACL*, pp. 4487–4496, 2019.

[16] I. Misra, A. Shrivastava, A. Gupta, and M. Hebert, "Cross-stitch networks for multi-task learning," in *CVPR*, pp. 3994–4003, 2016.

[17] S. Ruder, J. Bingel, I. Augenstein, and A. Søgaard, "Latent multi-task architecture learning," in *AAAI*, pp. 4822–4829, 2019.

[18] A. Søgaard and Y. Goldberg, "Deep multi-task learning with low level tasks supervised at lower layers," in *ACL*, 2016.

[19] K. Hashimoto, C. Xiong, Y. Tsuruoka, and R. Socher, "A joint many-task model: Growing a neural network for multiple NLP tasks," in *EMNLP*, pp. 1923–1933, 2017.

[20] C. Devin, A. Gupta, T. Darrell, P. Abbeel, and S. Levine, "Learning modular neural network policies for multi-task and multi-robot transfer," pp. 2169–2176, 05 2017.

[21] F. Alet, T. Lozano-Perez, and L. P. Kaelbling, "Modular meta-learning," in *Proceedings of The 2nd Conference on Robot Learning* (A. Billard, A. Dragan, J. Peters, and J. Morimoto, eds.), vol. 87 of *Proceedings of Machine Learning Research*, pp. 856–868, PMLR, 29–31 Oct 2018.

[22] C. Ying, A. Klein, E. Christiansen, E. Real, K. Murphy, and F. Hutter, "NAS-bench-101: Towards reproducible neural architecture search," in *Proceedings of the 36th International Conference on Machine Learning* (K. Chaudhuri and R. Salakhutdinov, eds.), vol. 97 of *Proceedings of Machine Learning Research*, pp. 7105–7114, PMLR, 09–15 Jun 2019.

[23] C. White, S. Nolen, and Y. Savani, "Exploring the loss landscape in neural architecture search," 2021.

[24] A. Zela, J. N. Siems, L. Zimmer, J. Lukasik, M. Keuper, and F. Hutter, "Surrogate NAS benchmarks: Going beyond the limited search spaces of tabular NAS benchmarks," in *International Conference on Learning Representations*, 2022.

[25] Y. Bengio, N. Léonard, and A. Courville, "Estimating or propagating gradients through stochastic neurons for conditional computation," *arXiv preprint arXiv:1308.3432*, 2013.

[26] M. Courbariaux, Y. Bengio, and J.-P. David, "Binaryconnect: Training deep neural networks with binary weights during propagations," 2015.

[27] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, and Y. Bengio, "Binarized neural networks," in *Advances in Neural Information Processing Systems* (D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, eds.), vol. 29, Curran Associates, Inc., 2016.

[28] E. F. T. K. Sang and F. D. Meulder, "Introduction to the conll-2003 shared task: Language-independent named entity recognition," in *HLT-NAACL*, pp. 142–147, 2003.

[29] X. Ma and E. H. Hovy, "End-to-end sequence labeling via bi-directional lstm-cnns-crf," in *ACL*, 2016.

[30] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings* (Y. Bengio and Y. LeCun, eds.), 2015.

[31] X. Dong and Y. Yang, "Nas-bench-201: Extending the scope of reproducible neural architecture search," in *International Conference on Learning Representations*, 2020.

초록

최근의 딥러닝의 발전이 컴퓨터 비젼, 자연어처리, 과학적 발견과 같은 복잡한 문제를 풀 수 있게 하였지만, 이와 같은 고품질의 결과는 여전히 거대한 양의 계산량에 의존하고 있다. 신경망 구조의 가용성을 늘리기 위한 하나의 방법으로, 하나의 슈퍼네트워크에 다중 작업을 학습시켜 작업간 공유할 수 있는 정보를 활용하는 방법을 사용해볼 수 있다. 다중 작업 학습의 최근 연구에서 듬성한 공유 방법은 기존의 다중 학습의 공유 방법에 비해 파라미터 효율과 역전이효과의 감소 측면에서 좋은 결과를 보여주었지만, 기존 듬성한 방법은 각 작업을 위한 모듈화 규칙을 찾는 계산 비용이 매우 비싸다는 한계가 있다. 이 논문에서는 미분 가능한 모듈화 학습을 도입해 각 작업을 위한 모듈화 규칙을 찾는 비용을 완화시키려고 한다. 이 방법은 연속 도메인 이완을 이용해 한 번의 학습으로 모든 작업에 대한 모듈화 규칙을 찾아낸다. 우리는 이 방법을 CoNLL-2003 데이터셋의 세가지 문자열 라벨링 작업에 대해 예증하였고, 평균 정확도는 기존의 작업과 거의 비슷한 수준을 얻지만 모듈화 규칙은 더욱 빠르게 찾는 것을 보였다. 마지막으로 우리는 모듈화 규칙의 밀집 정도에 따른 성능 저하와 모듈화 규칙간의 겹치는 비율을 분석하였다.

주요어: 다중 작업 학습, 모듈화 학습, 뉴럴 구조 탐색
학번: 2020-23215