Pruning neural network via reinforement learning

Han Zhang hz5g21@soton.ac.uk

1 Introduction

With the emergence of personal devices and SNS, more and more data is generated from centralized device data to local personal devices. To accommodate this, machine learning algorithms are being adapted to run locally, which can also improve user privacy, reduce latency, and increase energy efficiency. But large neural networks trained on centralized data cannot be directly deployed to small devices due to their numerous parameters and device constraints. This requires a leaner and more efficient neural network, which requires pruning the neural network[1].

Neural network pruning is a key technology to efficiently deploy neural network models to small devices under limited computing resources. This is a model optimization technique that involves removing redundant values from the weight tensor. The core of traditional pruning methods is to determine the compression strategy for each layer, as they have various degrees of redundancy, but this often requires hand-crafted heuristics and domain experts to trade-off between model size, speed, and accuracy. For example, fewer parameters are pruned in the first layer since the extracted features have a lower level and the least number of parameters, but more parameters are removed in the FC layer since the FC layer has the most parameters, and the layers that are sensitive to pruning are removed Fewer parameters, etc [2]. However, since the layers of deep neural networks are not independent, these rule-based pruning strategies are usually non-optimal.

Human heuristics are often sub-optimal and manual model compression is time-consuming. For this reason, automatic pruning of neural networks via reinforcement learning has significant advantages [3]. By automating this process through reinforcement learning, rather than relying on rule-based policies and experience, pruned neural networks are transferable and are often optimal.

2 Survey

For the neural network pruning via reinforcement learning, is the process that which an extensive neural network is replaced by a pruned network. This process is defined as a Markov decision process (MDP). The environment represents the network architecture, while the action is limited to the pruning of this large neural network, and finally, the similarity of the model is obtained as the error function for optimization so that the RL-based pruning neural network can be obtained.

2.1 Policy gradient

The first method is the policy gradient algorithm[4]. The authors proposed a method by replacing a large "teacher" network with a compressed "student" network. This "teacher"

network is a large neural network trained on centralized data. The "student" network is a small network after pruning. The model structure is shown in Figure 1.

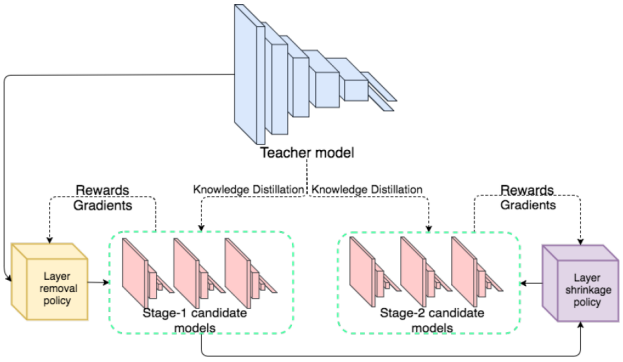

Figure 1: Model structure of pruning policy[4]

The policy of the first loop is that the network actively removes layers from the large "teacher" model. Then, another recurrent policy network carefully reduces the size of each remaining layer. The resulting network is then evaluated for a reward. The reward is based on model parameter reduction and the two models maintain similar outputs, and our method uses this reward signal with policy gradients to train the policy to find a locally optimal student network. Experiments show that the models perform well on VGG, ResNet, etc.

Strengths and weaknesses:This method achieves the results of automatically pruning neural networks but it has one serious drawback. The process of transforming a "teacher" network into a "student network" is defined as a Markov Decision Process. Under this model, states represent the network architecture. The domain of state S is very large since it contains every possible simplified structure of the "teacher" network. The reward function is defined due to the precision and compression ratio of the specified architecture. Applying reinforcement learning directly to solve this problem can be very slow.

2.2 Deep deterministic policy gradient

Google, along with MIT and Carnegie Mellon University[2] proposed an automatic model compression algorithm (AutoML for Model Compression, AMC) in 2018. It uses reinforcement learning to automatically search the design space, which greatly improves the compression of the model. Utilizing deep deterministic policy gradient (DDPG) agents can encourage model shrinkage and speedup. The actor-critic structure also helps reduce variance and promotes more stable training.

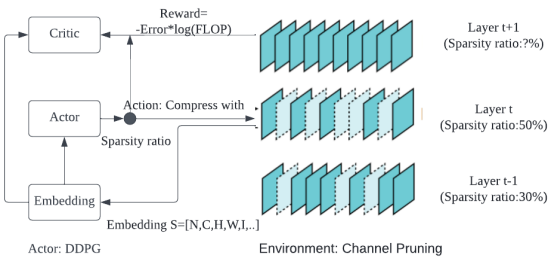

Figure 2: Structure of AMC model[2]

The structure of the model is demonstrated in Figure 2. The model mainly adopts the agent(DDPG) to receive the embedding s_t from layer t, then encodes the useful features of this layer, and finally outputs the exact compression ratio a_t. After layer t is compressed at a_t, the agent moves to the next layer $t+1$. After all the layers are compressed, the accuracy of the model will be calculated. Finally, the function of accuracy, FLOP(Floating-point operations per second) and reward R will return to the agent.

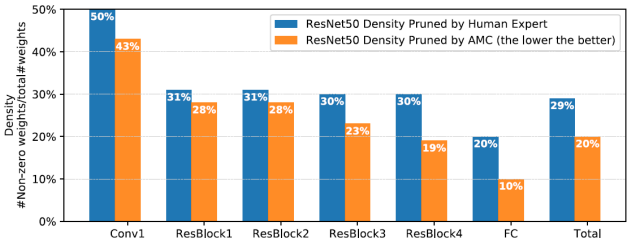

Figure 3: AMC compared with hand-designed heuristic strategy[2]

They propose two compression strategies. One is to achieve the best accuracy and the other is to achieve the smallest model in 2 different situations. The model is evaluated on VGG, ResNet, and MobileNet. The result shows that the AMC has better performance than the hand-designed heuristic strategy, shown in Figure 2. For ResNet50, it increases the expert-tuned compression ratio from 3.4× to 5.0× without loss of accuracy.

Strengths and weaknesses:This method achieves better than traditional rule-based compression strategies with higher compression ratios and better accuracy. However, since the model only gives rewards at the end of each iteration, the training period and the number of training times are greatly increased, making the model training slow.

2.3 Deep Q-Network

Based on AMC research, Manas and co-authors[5] propose a pruning model(PuRL). Compared with AMC, PuRL prunes based on different Markov Decision processes (MDP). An important part of this model is to provide "dense rewards" instead of relying on "sparse rewards" which are given only at the end of each episode. This results in significantly shorter training cycles, causing 85% fewer training episodes shown in Table 1.

	Sparsity	Starting_Acc	Pruned_Acc	RL Episodes
AMC	80%	76.13%	76.11%	400
RuRL	80.27%	76.13%	75.31%	55

Table 1: RuRL compared with the AMC model

The model structure is shown in Figure 4, which uses Deep Q-Network as the agent. PuRL assigns each layer a unique compression ratio α. Then, after pruning the layer, it gets feedback on test accuracy and sparsity. The difference with AMC is that the retraining of AMC is done after all layers are pruned, not after each layer, so the network cannot directly infer how the accuracy is. This may enlarge the training periods because more samples are required to infer the final accuracy of the network for each layer. However, PuRL tries to give rewards after removing each layer of content, rather than giving rewards at the end of chapters. This approach of giving dense rewards helps to achieve convergence faster than giving sparse rewards.

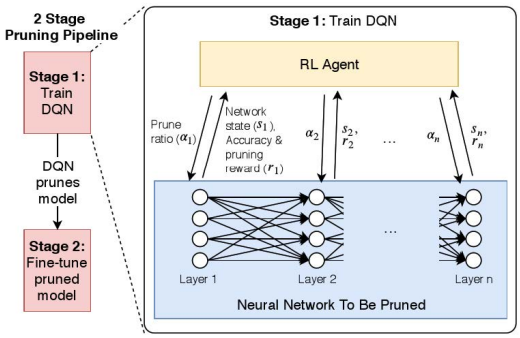

Figure 4: Structure of PuRL model

Strengths and weaknesses:This method solves the shortcoming that the AMC model cannot converge quickly but has a corresponding impact on the accuracy of the pruned neural network.

3 Analysis

In order to deploy the deep neural network algorithms to small devices. Neural network pruning has been a hot topic for a long period. It has been many years from the initial community lacking standardized benchmarks and metrics to the now increasingly explicit use of reinforcement learning for neural network pruning. From random pruning at the beginning to manual pruning through expert experience, at the end delete more parameters at the FC layer because it has the most parameters, at the beginning, delete fewer layers because the extraction level is low, until now automated pruning through reinforcement learning. The most cutting-edge is the pruning neural network via reinforcement learning.

3.1 State of the art

Previously, Neural network pruning methods are mostly random pruning and manual pruning based on the experts' experience, but those methods lack a standard and are inconsistent between papers so it is hard to compare the quality of various methods.

Ashok(2017) [4] attempted to apply reinforcement learning to neural network pruning and compare the results of other methods on different neural network constructions. The results show that algorithms adopting Policy gradient-based Actor-Critic outperform most manual pruning methods, and on ResNet34, more than 10× compression ratio can be achieved, at the same time, the compressed model can maintain similar performance.

He and colleagues(2018) [2] argue that traditional model compression techniques using hand-crafted features require domain experts to explore large design space and trade-off model size, speed, and accuracy, which are often suboptimal, and require a lot of experts to design. They proposed the AutoML for Model Compression (AMC) algorithm, which uses reinforcement learning to automatically search the design space, greatly improving the quality of model compression. The models demonstrate convincing results on neural network models such as ResNet and VGG. The compressed model has good generalization from classification tasks to detection tasks. Improves the MobileNet inference speed from 8.1 fps to 16.0 fps on the deployment mobile phone running speed. This model greatly facilitates efficient deep neural network design on mobile devices.

Based on the AMC model, Liu and co-authors[6] believe that most current channel pruning methods (AMC) usually use a fixed compression ratio for all layers of the model, but this

method may not be optimal. They propose that given a specific target compression ratio, the optimal compression ratio for each layer can be searched by some automatic methods. To address this issue, they propose a Conditional Automatic Channel Pruning (CACP) method, which simultaneously produces compressed models at different compression ratios through a single channel pruning process. Extensive experiments on the CIFAR-10 and ImageNet datasets demonstrate the superiority of our method over existing methods, saving nearly 3 times the time compared to the AMC model.

At the same time, Manas [5]. propose a pruning model(PuRL). Compared with AMC, PuRL prunes based on different Markov Decision processes (MDP). An important part of this model is to provide "dense rewards" instead of relying on "sparse rewards" which are given only at the end of each episode. This results in significantly shorter training cycles, causing 85% fewer training episodes.

Malik [7] proposed a new pruning strategy to implement neural network pruning through a constrained reinforcement learning algorithm. On VGG, the model reduces 83 - 92.90% of the total parameters while achieving comparable or better performance than the original network. On ResNet18, the model reduces parameters by 75.09% without any accuracy loss. Compared to CACP, CRL achieves a nearly 4-fold parameter reduction but a 0.8% drop inaccuracy on VGG16. Compared with AMC, CRL achieves a parameter reduction of 3 million and an accuracy increase of 0.6% on Resnet50.

3.2 Open problem

Because model compression and neural network pruning change from a large Full-order neural network("teacher" network) to a Pruned neural network("student" network)[8]. The main disadvantage of pruning is that most deep learning frameworks and hardware cannot accelerate the computation of sparse matrices, which means that regardless of the parameter matrix No matter how sparse, it doesn't have a substantial impact on the actual training cost[9].

On this basis, the standards for generating pruned neural networks are also different. Generally speaking, there are fewer parameters, and less training time and the network generates similar results, making it challenging to define an excellent pruned neural network.

4 Conclusion

In this research, I learned about the cutting-edge application of reinforcement learning and the advantages of reinforcement learning in neural network pruning. Different pruning effects can be achieved by setting different actors and rewards. For example, changing the sparse reward in AMC to the dense reward in PuRL can speed up the model. Also, different policy algorithms can lead to slightly different results, such as Malik's CRL model outperforming CACP and AMC models in the direction of parameter reduction. But at the same time, due to the different standards of model compressions, such as fewer parameters, faster training, and similar results, leads to the fact that no absolute algorithm is better than other algorithms, so it is necessary to compare algorithms on different datasets.

For future work, I found that the existing literature mainly focuses on the compression of convolutional neural networks and fully connected layers. At the same time, there is little work on the compression of recurrent neural networks[10]. In the future, the compression work of the recurrent neural network can be realized.

References

[1] Jiaqi Li, Ross Drummond, and Stephen R. Duncan. Robust error bounds for quantised and pruned neural networks. 11 2020.

[2] Yihui He, Ji Lin, Zhijian Liu, Hanrui Wang, Li-Jia Li, and Song Han. Amc: Automl for model compression and acceleration on mobile devices. 2 2018.

[3] Ning Liu, Xiaolong Ma, Zhiyuan Xu, Yanzhi Wang, Jian Tang, and Jieping Ye. Autocompress: An automatic dnn structured pruning framework for ultra-high compression rates. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 4876–4883, 2020.

[4] Anubhav Ashok, Nicholas Rhinehart, Fares Beainy, and Kris M. Kitani. N2n learning: Network to network compression via policy gradient reinforcement learning. 9 2017.

[5] Manas Gupta, Siddharth Aravindan, Aleksandra Kalisz, Vijay Chandrasekhar, and Lin Jie. Learning to prune deep neural networks via reinforcement learning. 7 2020.

[6] Yixin Liu, Yong Guo, Jiaxin Guo, Luoqian Jiang, and Jian Chen. Conditional automated channel pruning for deep neural networks. *IEEE Signal Processing Letters*, 28:1275–1279, 2021.

[7] Shehryar Malik, Muhammad Umair Haider, Omer Iqbal, and Murtaza Taj. Neural network pruning through constrained reinforcement learning. 10 2021.

[8] Ross Drummond, Mathew C. Turner, and Stephen R. Duncan. Reduced-order neural network synthesis with robustness guarantees. 2 2021.

[9] Torsten Hoefler, Dan Alistarh, Tal Ben-Nun, Nikoli Dryden, and Alexandra Peste. Sparsity in deep learning: Pruning and growth for efficient inference and training in neural networks. *Journal of Machine Learning Research*, 22(241):1–124, 2021.

[10] Rahul Mishra, Hari Prabhat Gupta, and Tanima Dutta. A survey on deep neural network compression: Challenges, overview, and solutions. *arXiv preprint arXiv:2010.03954*, 2020.