Train-free NAS for RNN

Andrei Filatov [1]

Abstract

Neural architecture search (NAS) is a technique for automating the design of neural networks, a widely used model in the field of machine learning. But there are two problem to address. The first problem is that huge number of computational resources to train optimal model. The second is focus on non-sequential models. The majority of methods is created for non-sequential models. To address both problems we will try to find the optimal heuristic for recurrent models which allows assess the quality of model without or with small training. The experiments will be conducted on NAS-Bench-NLP.

The code and work is available on https://github.com/anvilarth/nas-rnn.

1. Problem statement

Selecting proper architecture for a certain task is hard problem. Researchers spend dozen of time manually adjusting architectures to achieve good results. Neural architecture search (NAS) was introduced to resolve this problem through automatization architecture's selection process. NAS allow to find the optimal architecture from predefined set of possible architectures. This property is utilized when one want to maximize performance conditioned by computational budget or to satisfy inference condition. The good example of it is EfficientNets (Tan & Le, 2019) which achieved good results in comparison to classical ResNet architectures. But, NAS method requires huge amount of computational resources. To train one method on single it requires a hundreds of GPU hours. The second problem is that NAS methods is primarly focused on non-sequential models.

[*]Equal contribution [1]Skolkovo Institute of Science and Technology. Correspondence to: Andrei Filatov <andrei.filatov@skoltech.ru>.

Proceedings of the 38th International Conference on Machine Learning, PMLR 139, 2021. Copyright 2021 by the author(s).

2. Method

The idea of the following work is to adapt train-free methods to recurrent architecture. We will check applicability already proposed methods. To assess the quality of train-free method we will utilize NAS-Bench-NLP. This benchmark consists of 14k+ NLP architectures with known performance. We will their results to investigate the correlation between train-free heuristics and final performance. We will investigate Zen-score (Lin et al., 2021) and TE-NAS (Chen et al., 2021) as train-free criteria for assesing RNN models performance. The advantages of this idea is making more accessible NAS for recurrent models. The possible disadvantage is that train-free methods may don't work on recurrent models and provide poor performance with respect to methods with training.

3. Related work

There are several types of algorithm which are utilized in Neural architecture search. The first approach is random search. In random search we just sample different architectures and check their performance. The second approach is to use evolutionary algorithms. The idea of these algorithms first create some population of networks. Then, mutate them and leave only best performance and repeat this procedure until achieving good performance. like AmoebaNet (Shah et al., 2018), EcoNAS (Zhou et al., 2020), CARS (), GeNet (Yang et al., 2020) and PNAS (Liu et al., 2018a). Also, there are reinforcement learning, which works in the following fashion. One sample the architecture, train and evaluate it, after it we use reward to update sampling model approaches. like NASNet (Zoph et al., 2018), Mnasnet (Tan et al., 2019) and MetaQNN (Baker et al., 2016). But these approaches are too computationally expensive so the hypernet or supernet here appears. Their idea is to predefine set of modules and optimize only parameters responsible to merging them in one network. The list of current approaches are DARTS (Liu et al., 2018b), SNAS (Xie et al., 2018), PC-DARTS (Xu et al., 2019), ProxylessNAS (Cai et al., 2018), GDAS (Dong & Yang, 2019), FBNetV2 (Wan et al., 2020), Single-Path One-Shot NAS (Guo et al., 2020).

Application of neural architecture search is more focused onto non-sequential model, which take only one object and provides only single output. The (Klyuchnikov et al., 2020) tried to solve this problem.

The classical neural architecture search utilizes vast amount of resources and researchers trying to minimize this spending. The idea is to create some proxy function which correlates with the performance and this proxy requires less resources to optimize than original function. The example can be a training model with less epochs and parameters (Abdelfattah et al., 2021). Or try to prune the initial model. The ultimate case of this idea is train-free approaches. The idea of train-free approaches is choose the optimal architecture without any training. This allows to increase the speed of training very fast. The train-free methods operates on some heuristics which correlates the final performance. The first is Zen-score (Lin et al., 2021). They proposed to use gradient norm as score for neural network. But without batch-normalization layers deep neural networks has numerical flow problems. So propose to use batch normalizations and change the way how the score is calculated. The second work is TE-NAS (Chen et al., 2021). They utilizes Neural Tangent Kernel parameters as condition number of NTK the number of linear regions to assess trainability and expressivity correspondingly

4. Experiment

To assess the quality of train-free scores we utilize NAS-Bench-NLP (Klyuchnikov et al., 2020). We use their code to reproduce some networks and utilize their test losses as a metric. Then, we calculated ZeNAS score and GradScore on these architectures.

$$Zen(f) = \log(\mathbb{E}_{\mathbf{x},\epsilon}\|f(\mathbf{x}) - f(\mathbf{x}+\alpha\epsilon)\|_F)$$

$$GradScore(f) = \mathbb{E}_{\mathbf{x}}\|\nabla(\mathbf{x})\|_F$$

On fig 1 and fig. 2 one can see that ZeNAS score has a positive correlation with test loss. Gradscore has no correlation with test loss. Which means that zenas score can be utilize as proxy for test performance. Whereas, gradscore gives no clue to test performance

5. Conclusion

TBD

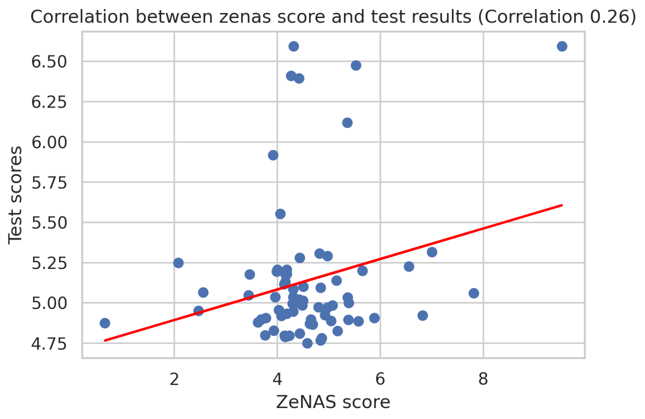

Figure 1. Scatterplot for zenas score and test losses

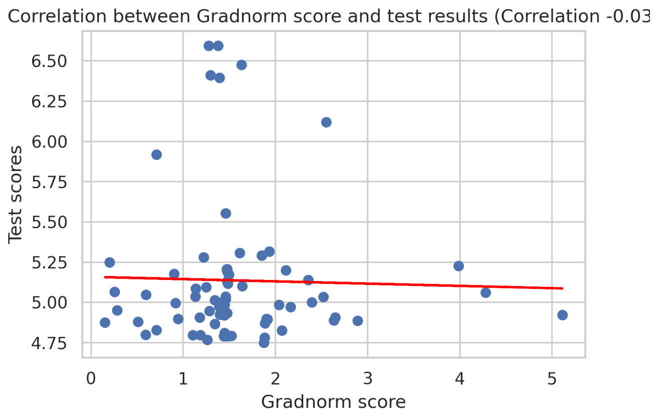

Figure 2. Scatterplot for gradnorm score and test losses

References

Abdelfattah, M. S., Mehrotra, A., Dudziak, Ł., and Lane, N. D. Zero-cost proxies for lightweight nas. *arXiv preprint arXiv:2101.08134*, 2021.

Baker, B., Gupta, O., Naik, N., and Raskar, R. Designing neural network architectures using reinforcement learning. *arXiv preprint arXiv:1611.02167*, 2016.

Cai, H., Zhu, L., and Han, S. Proxylessnas: Direct neural architecture search on target task and hardware. *arXiv preprint arXiv:1812.00332*, 2018.

Chen, W., Gong, X., and Wang, Z. Neural architecture search on imagenet in four gpu hours: A theoretically inspired perspective. *arXiv preprint arXiv:2102.11535*, 2021.

Dong, X. and Yang, Y. Searching for a robust neural architecture in four gpu hours. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 1761–1770, 2019.

Guo, Z., Zhang, X., Mu, H., Heng, W., Liu, Z., Wei, Y., and Sun, J. Single path one-shot neural architecture search with uniform sampling. In *European Conference on Computer Vision*, pp. 544–560. Springer, 2020.

Klyuchnikov, N., Trofimov, I., Artemova, E., Salnikov, M., Fedorov, M., and Burnaev, E. Nas-bench-nlp: neural architecture search benchmark for natural language processing. *arXiv preprint arXiv:2006.07116*, 2020.

Lin, M., Wang, P., Sun, Z., Chen, H., Sun, X., Qian, Q., Li, H., and Jin, R. Zen-nas: A zero-shot nas for high-performance deep image recognition. *arXiv preprint arXiv:2102.01063*, 2021.

Liu, C., Zoph, B., Neumann, M., Shlens, J., Hua, W., Li, L.-J., Fei-Fei, L., Yuille, A., Huang, J., and Murphy, K. Progressive neural architecture search. In *Proceedings of the European conference on computer vision (ECCV)*, pp. 19–34, 2018a.

Liu, H., Simonyan, K., and Yang, Y. Darts: Differentiable architecture search. *arXiv preprint arXiv:1806.09055*, 2018b.

Shah, S. A. R., Wu, W., Lu, Q., Zhang, L., Sasidharan, S., DeMar, P., Guok, C., Macauley, J., Pouyoul, E., Kim, J., et al. Amoebanet: An sdn-enabled network service for big data science. *Journal of Network and Computer Applications*, 119:70–82, 2018.

Tan, M. and Le, Q. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International conference on machine learning*, pp. 6105–6114. PMLR, 2019.

Tan, M., Chen, B., Pang, R., Vasudevan, V., Sandler, M., Howard, A., and Le, Q. V. Mnasnet: Platform-aware neural architecture search for mobile. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 2820–2828, 2019.

Wan, A., Dai, X., Zhang, P., He, Z., Tian, Y., Xie, S., Wu, B., Yu, M., Xu, T., Chen, K., et al. Fbnetv2: Differentiable neural architecture search for spatial and channel dimensions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 12965–12974, 2020.

Xie, S., Zheng, H., Liu, C., and Lin, L. Snas: stochastic neural architecture search. *arXiv preprint arXiv:1812.09926*, 2018.

Xu, Y., Xie, L., Zhang, X., Chen, X., Qi, G.-J., Tian, Q., and Xiong, H. Pc-darts: Partial channel connections for memory-efficient architecture search. *arXiv preprint arXiv:1907.05737*, 2019.

Yang, Z., Wang, Y., Chen, X., Shi, B., Xu, C., Xu, C., Tian, Q., and Xu, C. Cars: Continuous evolution for efficient neural architecture search. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 1829–1838, 2020.

Zhou, D., Zhou, X., Zhang, W., Loy, C. C., Yi, S., Zhang, X., and Ouyang, W. Econas: Finding proxies for economical neural architecture search. In *Proceedings of the IEEE/CVF Conference on computer vision and pattern recognition*, pp. 11396–11404, 2020.

Zoph, B., Vasudevan, V., Shlens, J., and Le, Q. V. Learning transferable architectures for scalable image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 8697–8710, 2018.