Bag of Tricks for Neural Architecture Search

Thomas Elsken[1], Benedikt Staffler[1], Arber Zela[2], Jan Hendrik Metzen[1] and Frank Hutter[2,1]
[1]Bosch Center for Artificial Intelligence, [2]University of Freiburg
{thomas.elsken, benediktsebastian.staffler, janhendrik.metzen}@de.bosch.com
{zelaa, fh}@cs.uni-freiburg.de

Abstract

While neural architecture search methods have been successful in previous years and led to new state-of-the-art performance on various problems, they have also been criticized for being unstable, being highly sensitive with respect to their hyperparameters, and often not performing better than random search. To shed some light on this issue, we discuss some practical considerations that help improve the stability, efficiency and overall performance.

1. Introduction

Neural architecture search (NAS) methods have been very successful in previous years and led to a new state of the art on various problems and benchmarks, e.g., for image classification [58, 35], semantic segmentation [26] or object detection [17]; please refer to the surveys [14, 46] for an overview. However, they have also been criticized for being unstable and for providing unfair or non-transparent empirical comparisons due to using various tweaks for boosting performance beside just comparing the optimized architecture, see, e.g., [25]. In particular methods employing one-shot models have been reported to be brittle, highly sensitive with respect to their hyperparameters, and often no better than random search [24, 52, 49, 53, 54, 50]. Tricks for stabilizing the search are often hidden in the details and are hard to find for the reader, or are not even discussed. In this short paper, we provide some insights in the details of using NAS methods and discuss common practices in NAS that help improving the stability (Section 2), efficiency (Section 3), and overall performance (Section 4).

2. Stabilizing Gradient-Based NAS and Training One-Shot Models

Weights Warm-Up. Gradient-based NAS methods typically employ a continuous relaxation of the architecture search space by considering a weighted combination of operations (such as convolution or pooling layers) [28]. This allows to search for architectures by using alternating stochastic gradient descent, which (in each batch) iterates updates of the network parameters and the real-valued weights parameterizing the architecture. However, directly using this alternating optimization has been reported to lead to premature convergence in the architectural space [26]. Consequently, a common trick [26, 38, 33, 49, 15, 17] is to start by optimizing the network weights only, often for as long as half of the overall search epochs; architecture updates are only conducted afterwards. This trick is important in order for the architecture search to not favour architectures that train faster (in particular those that contain many skip connections).

Similar approaches for warming up weights can be found for sampling-based methods. Bender et al. [2] start by training the whole one-shot model and then drop out more and more paths over the course of training. TuNAS [3] adapts this strategy; while they directly samples paths from the one-shot models for training, they enable all operations within a certain block of the one-shot model rather than only the sampled operation. The probability for enabling all operations is annealing to 0 over the course of training.

It is even a possibility to first *fully* train the one-shot model and conduct the search afterwards, thus decoupling these two stages [2, 18, 7].

Regularization and Loss Landscape Smoothing. It was shown [53] that smoothing the loss landscape by using stronger regularization can help to stabilize architecture search. This can, e.g., be done via stochastic regularization techniques, such as drop path [58], weight decay or data augmentation. Alternatively, more robust loss functions can achieve a similar goal, e.g., by minimizing the loss in a neighbourhood of an optimal architecture rather than only for the optimum [6] or by implicitly smoothing the loss function via additional auxiliary connections [8].

Normalization layers. For NAS methods using a continuous relaxation of the search space, such as DARTS, a naive use of normalization layers such as batch [22], layer [1], instance [42] or group [47] normalization, is problematic since their learnable parameters can lead to a rescaling of the architectural parameters and thus make them meaning-

less. Consequently, the learnable parameters are typically disabled [28]. Xu et al. [49] even report that in general batch normalization was harmful in their experiments and hence they do not use it at all. Furthermore, batch normalization can cause issues in combination with NAS methods that require to keep the one-shot model in memory since this naturally leads to using small batch sizes due to memory limitations. This is especially problematic for applications with high-resolution input images.

Some normalization layers are also fundamentally problematic in combination with sampling-based methods since the normalization statistics will vary across different sampled paths. Bender et al. [2] report that training the one-shot models was highly unstable in early stages of experimentation, and that these instabilities were overcome by using batch statistics also during evaluation and a variant of ghost batch normalization [19]. Many researchers also replace standard batch normalization by more advanced techniques, e.g., [7] use synchronized batch normalization [32] across GPUs to increase the effective batch size and recalculate batch statistics during architecture optimization and [44] use group normalization [47] instead. We also refer to [10] for a discussion of batch normalization within models trained by sampling paths.

3. Speeding Up NAS

Proxy Tasks. A very common approach for speeding up NAS is to use lower fidelity (or proxy) estimates. E.g., approaches using a cell-based search space typically use fewer cells with fewer filters during search than during evaluation [58] and train for fewer epochs. The size of the trainig data set can also be reduced to make the search more efficient, e.g., by downscaling images [26] or by searching on a smaller data set (e.g., CIFAR or PennTreeBank) and transferring the learned cells to a larger one (e.g. ImageNet or WikiText-2) as is often done in practice [58, 35, 28]. We refer to Elsken et al. [14] for a general overview. Zhou et al. [57] study the impact of such lower fidelity estimates and assess how different proxies should be used in combination to achieve the best speed up while maintaining a high correlation with the true optimization metric.

Feature Caching. Recently, many researchers have applied NAS methods to tasks such as semantic segmentation [26] or object detection [49], where architectures are composed of several components, such as a backbone and a task-specific head. When the backbone is fixed during the search, its outputs can be *pre-computed* once for all training data points to avoid unnecessary computation and thereby speed up architecture search [5, 31, 44].

Speeding Up The Optimization Process via Sequential Search. Rather than optimizing different components of an architecture jointly, the search is often split up into several phases for different components in order to reduce memory and time consumption. For example, in the case of object detection, Xu et al. [49] first search for the multi-scale feature extractor and then for the detection head. Du et al. [13] first search for scale permutations of a given network and then tune the building blocks of the resulting architecture, e.g., by adjusting the resolution of feature maps and by choosing one out of a set of predefined possible building blocks, such as a residual block or a bottleneck block. Guo et al. [17] first sequentially screen different search spaces for different architectural components with a downscaled model and prune the search spaces before conducting a final optimization of the reduced search spaces.

Pre-Optimized Search Spaces. While in principle NAS can be viewed as a subfield of automated machine learning (AutoML) [21] and thus aims for searching for architectures with as little prior knowledge from humans as possible, it can nevertheless be helpful to build search spaces around architectures that are known to work well for efficiency reasons, rather than searching from scratch [36]. For example, search spaces are often based on inverted residual blocks [39], essentially resulting in optimizing the hyperparameters that come with these blocks, such as kernel sizes, expansion ratios or dilatation rates [40, 17, 3, 4]. Some methods also directly build upon existing architectures and search for transformations of these architectures, e.g., via permuting layers [13] or by searching how to connect channel groups within an architecture [33]. We note that while this use of pre-optimized search spaces is likely to yield improved results for a particular application more quickly, this process cannot discover entirely new architectures, such as Transformer [43] architectures. In order to achieve the latter, one would have to use dramatically more powerful search spaces, and potentially with a hierarchical structure [27, 48, 37].

4. Improving the Final Performance

Deriving Optimal Architectures from the Search Process. Identifying the optimal architectures from NAS runs is not trivial for at least the following reasons: Firstly, as almost all methods employ lower fidelity estimates, the ranking of architectures on the proxy tasks will likely be different from the ranking on the true task. Secondly, it is currently not well understood how weight sharing affects the ranking of architectures. Some researchers show that weight sharing is not necessarily properly ranking architectures [24, 52, 50, 54]. Consequently, researchers often first collect a set of candidate architectures, either by running NAS multiple times [28] or by obtaining multiple architectures from a single run of the method (e.g., by sampling from a learned distribution or by sampling from a population of evolved networks) [58, 5]. These sets of candidate architectures are then evaluated in a setting which has higher correlation with respect to the setting of interest and

the best out of the candidates is chosen to be the optimal architecture. This process is sometimes also already used within the search process when components are searched sequentially [44]. To increase correlation between ranking of architectures with weights inherited from the one-shot model versus when retrained from scratch, Zhao et al. [56] propose to use a set of sub-one-shot models, where each sub model covers different regions of the search space, with the goal of alleviating undesired co-adaption. Additionally, for approaches employing a continuous relaxation of the search space, it remains unclear what the best way is to obtain a discretized architecture from the real-valued parameterization. Typically, the operations with maximum weight are chosen as initially proposed by Liu et al. [28]. Wang et al. [45] argue that this process is suboptimal since the operation weights are not directly correlated with performance of the resulting architecture and thus propose a different scheme for extracting a discretized architecture based on minimizing the drop in performance when removing an operation from the one-shot model.

Hyperparameters, Data Augmentation and other Tweaks for Boosting Performance. The performance of a neural architecture depends on many factors other than the architecture itself, such as data augmentation [11, 55, 9], stochastic regularization [16, 58], activation functions [34] and other hyperparameters such as learning rate (schedules) [29]. Yang et al. [50] provide a thorough ablation study on these factors on CIFAR-10. They show that the training pipeline is more important than the architecture: The worst out of eight randomly sampled architectures trained with the best training pipeline substantially outperformed the best of the eight architectures using the worst training pipeline. To give another example, MobileNetV3 [20] achieved 75.2% top-1 accuracy on ImageNet, suggesting an improvement of 3.2% due to the novel architecture compared to the performance of 72.0% for MobileNetV2 [39]. However, Bender et al. [3] show that when both models are trained with an identical state-of-the-art training pipeline, MobileNetV2 achieves 73.3% accuracy compared to 75.3% for MobileNetV3, thereby reducing the improvement due to the architecture from 3.2% to 2.0%. Thus, all these factors along with the architecture heavily impact the final performance. Moreover, the search hyperparameters are in particular important for one-shot NAS methods as already discussed above. Zela et al. [54] optimize the hyperparameters of various one-shot NAS algorithms and show that the found solutions can outperform black-box NAS optimizers when properly tuned. To avoid many of these confounding factors when comparing different NAS algorithms, a series of NAS benchmarks [51, 54, 12, 41, 30, 23] have been proposed.

5. Conclusion

We presented a list of tips and tricks for employing NAS methods and making them more robust in practice. We hope that these can ease the usability of NAS methods, both for experienced and new researchers.

References

[1] Lei Jimmy Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer normalization. *arXiV*, 2016. 1

[2] Gabriel Bender, Pieter-Jan Kindermans, Barret Zoph, Vijay Vasudevan, and Quoc Le. Understanding and simplifying one-shot architecture search. In *ICML*, 2018. 1, 2

[3] Gabriel Bender, Hanxiao Liu, Bo Chen, Grace Chu, Shuyang Cheng, Pieter-Jan Kindermans, and Quoc V. Le. Can weight sharing outperform random architecture search? an investigation with tunas. In *The IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020. 1, 2, 3

[4] Bo Chen, Golnaz Ghiasi, Hanxiao Liu, Tsung-Yi Lin, Dmitry Kalenichenko, Hartwig Adam, and Quoc V. Le. Mnasfpn: Learning latency-aware pyramid architecture for object detection on mobile devices. In *The IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020. 2

[5] Liang-Chieh Chen, Maxwell Collins, Yukun Zhu, George Papandreou, Barret Zoph, Florian Schroff, Hartwig Adam, and Jon Shlens. Searching for efficient multi-scale architectures for dense image prediction. In *NeurIPS 31*. 2018. 2

[6] Xiangning Chen and Cho-Jui Hsieh. Stabilizing differentiable architecture search via perturbation-based regularization. In *ICML*, pages 1554–1565. PMLR, 2020. 1

[7] Yukang Chen, Tong Yang, Xiangyu Zhang, Gaofeng Meng, Xinyu Xiao, and Jian Sun. Detnas: Backbone search for object detection. In *NeurIPS*. 2019. 1, 2

[8] Xiangxiang Chu, Xiaoxing Wang, Bo Zhang, Shun Lu, Xiaolin Wei, and Junchi Yan. {DARTS}-: Robustly stepping out of performance collapse without indicators. In *ICLR*, 2021. 1

[9] Ekin D. Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, and Quoc V. Le. Autoaugment: Learning augmentation strategies from data. In *CVPR*, June 2019. 3

[10] Zhijie Deng, Yinpeng Dong, Shifeng Zhang, and Jun Zhu. Understanding and exploring the network with stochastic architectures. In *NeurIPS 33*. 2020. 2

[11] Terrance DeVries and Graham W. Taylor. Improved regularization of convolutional neural networks with cutout. arXiv, 2017. 3

[12] Xuanyi Dong and Yi Yang. Nas-bench-201: Extending the scope of reproducible neural architecture search. In *ICLR*, 2020. 3

[13] Xianzhi Du, Tsung-Yi Lin, Pengchong Jin, Golnaz Ghiasi, Mingxing Tan, Yin Cui, Quoc V. Le, and Xiaodan Song. Spinenet: Learning scale-permuted backbone for recognition and localization. In *CVPR*, June 2020. 2

[14] Thomas Elsken, Jan Hendrik Metzen, and Frank Hutter. Neural architecture search: A survey. *Journal of Machine Learning Research*, 20(55):1–21, 2019. 1, 2

[15] Thomas Elsken, Benedikt Staffler, Jan Hendrik Metzen, and Frank Hutter. Meta-learning of neural architectures for few-shot learning. In *The IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020. 1

[16] Xavier Gastaldi. Shake-shake regularization. In *ICLR Workshop*, 2017. 3

[17] Jianyuan Guo, Kai Han, Yunhe Wang, Chao Zhang, Zhaohui Yang, Han Wu, Xinghao Chen, and Chang Xu. Hit-detector: Hierarchical trinity architecture search for object detection. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. 1, 2

[18] Zichao Guo, Xiangyu Zhang, Haoyuan Mu, Wen Heng, Zechun Liu, Yichen Wei, and Jian Sun. Single path one-shot neural architecture search with uniform sampling. In *European Conference on Computer Vision*, pages 544–560. Springer, 2020. 1

[19] Elad Hoffer, Itay Hubara, and Daniel Soudry. Train longer, generalize better: closing the generalization gap in large batch training of neural networks. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *NeurIPS 30*, pages 1731–1741. Curran Associates, Inc., 2017. 2

[20] Andrew Howard, Mark Sandler, Grace Chu, Liang-Chieh Chen, Bo Chen, Mingxing Tan, Weijun Wang, Yukun Zhu, Ruoming Pang, Vijay Vasudevan, Quoc V. Le, and Hartwig Adam. Searching for mobilenetv3. In *ICCV*, October 2019. 3

[21] Frank Hutter, Lars Kotthoff, and J. Vanschoren. *Automated Machine Learning: Methods, Systems, Challenges*. Challenges in Machine Learning. Springer, 2019. 2

[22] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the 32nd International Conference on Machine Learning*, 2015. 1

[23] Chaojian Li, Zhongzhi Yu, Yonggan Fu, Yongan Zhang, Yang Zhao, Haoran You, Qixuan Yu, Yue Wang, Cong Hao, and Yingyan Lin. {HW}-{nas}-bench: Hardware-aware neural architecture search benchmark. In *ICLR*, 2021. 3

[24] Liam Li and Ameet Talwalkar. Random search and reproducibility for neural architecture search. *Conference on Uncertainty in Artificial Intelligence (UAI)*, 2019. 1, 2

[25] Marius Lindauer and Frank Hutter. Best practices for scientific research on neural architecture search. *Journal of Machine Learning Research*, 21(243):1–18, 2020. 1

[26] Chenxi Liu, Liang-Chieh Chen, Florian Schroff, Hartwig Adam, Wei Hua, Alan L. Yuille, and Li Fei-Fei. Auto-deeplab: Hierarchical neural architecture search for semantic image segmentation. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019. 1, 2

[27] Chenchen Liu, Miao Hu, John Paul Strachan, and Hai Li. Rescuing memristor-based neuromorphic design with high defects. In *54th Annual Design Automation Conference (DAC)*, pages 1–6, 2017. 2

[28] Hanxiao Liu, Karen Simonyan, and Yiming Yang. DARTS: Differentiable architecture search. In *ICLR*, 2019. 1, 2, 3

[29] I. Loshchilov and F. Hutter. Sgdr: Stochastic gradient descent with warm restarts. In *ICLR*, 2017. 3

[30] Abhinav Mehrotra, Alberto Gil C. P. Ramos, Sourav Bhattacharya, Łukasz Dudziak, Ravichander Vipperla, Thomas Chau, Mohamed S Abdelfattah, Samin Ishtiaq, and Nicholas Donald Lane. {NAS}-bench-{asr}: Reproducible neural architecture search for speech recognition. In *ICLR*, 2021. 3

[31] Vladimir Nekrasov, Hao Chen, Chunhua Shen, and Ian Reid. Fast neural architecture search of compact semantic segmentation models via auxiliary cells. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019. 2

[32] Chao Peng, Tete Xiao, Zeming Li, Yuning Jiang, Xiangyu Zhang, Kai Jia, Gang Yu, and Jian Sun. Megdet: A large mini-batch object detector. In *CVPR*, pages 6181–6189, 2018. 2

[33] Junran Peng, Ming Sun, ZHAO-XIANG ZHANG, Tieniu Tan, and Junjie Yan. Efficient neural architecture transformation search in channel-level for object detection. In *NeurIPS 32*. 2019. 1, 2

[34] Prajit Ramachandran, Barret Zoph, and Quoc V. Le. Searching for activation functions, 2018. 3

[35] Esteban Real, Alok Aggarwal, Yanping Huang, and Quoc V. Le. Aging Evolution for Image Classifier Architecture Search. In *AAAI*, 2019. 1, 2

[36] Esteban Real, Chen Liang, David R So, and Quoc V Le. Evolving machine learning algorithms from scratch. *ICML*, 2020. 2

[37] Robin Ru, Pedro Esperança, and Fabio Maria Carlucci. Neural architecture generator optimization. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *NeurIPS*, volume 33, pages 12057–12069. Curran Associates, Inc., 2020. 2

[38] Tonmoy Saikia, Yassine Marrakchi, Arber Zela, Frank Hutter, and Thomas Brox. Autodispnet: Improving disparity estimation with automl. In *ICCV*, October 2019. 1

[39] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. MobileNetV2: Inverted residuals and linear bottlenecks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 2, 3

[40] Albert Shaw, Daniel Hunter, Forrest Landola, and Sammy Sidhu. Squeezenas: Fast neural architecture search for faster semantic segmentation. In *ICCV Workshops*, Oct 2019. 2

[41] Julien Siems, Lucas Zimmer, Arber Zela, Jovita Lukasik, Margret Keuper, and Frank Hutter. Nas-bench-301 and the case for surrogate benchmarks for neural architecture search. *arXiv preprint*, abs/2008.09777, 2020. 3

[42] Dmitry Ulyanov, Andrea Vedaldi, and Victor S. Lempitsky. Instance normalization: The missing ingredient for fast stylization. *arXiv*, abs/1607.08022, 2016. 1

[43] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *NeurIPS*, volume 30. Curran Associates, Inc., 2017. 2

[44] Ning Wang, Yang Gao, Hao Chen, Peng Wang, Zhi Tian, Chunhua Shen, and Yanning Zhang. Nas-fcos: Fast neural architecture search for object detection. In *The IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020. 2, 3

[45] Ruochen Wang, Minhao Cheng, Xiangning Chen, Xiaocheng Tang, and Cho-Jui Hsieh. Rethinking architecture selection in differentiable NAS. In *ICLR*, 2021. 3

[46] M. Wistuba, Ambrish Rawat, and Tejaswini Pedapati. A survey on neural architecture search. *ArXiv*, abs/1905.01392, 2019. 1

[47] Yuxin Wu and Kaiming He. Group normalization. In *Proceedings of the European Conference on Computer Vision (ECCV)*, September 2018. 1, 2

[48] Saining Xie, Alexander Kirillov, Ross Girshick, and Kaiming He. Exploring randomly wired neural networks for image recognition. In *ICCV*, October 2019. 2

[49] Hang Xu, Lewei Yao, Wei Zhang, Xiaodan Liang, and Zhenguo Li. Auto-fpn: Automatic network architecture adaptation for object detection beyond classification. In *ICCV*, October 2019. 1, 2

[50] Antoine Yang, Pedro M. Esperança, and Fabio M. Carlucci. Nas evaluation is frustratingly hard. In *ICLR*, 2020. 1, 2, 3

[51] Chris Ying, Aaron Klein, Eric Christiansen, Esteban Real, Kevin Murphy, and Frank Hutter. Nas-bench-101: Towards reproducible neural architecture search. In *ICML*, 2019. 3

[52] Kaicheng Yu, Christian Sciuto, Martin Jaggi, Claudiu Musat, and Mathieu Salzmann. Evaluating the search phase of neural architecture search. In *ICLR*, 2020. 1, 2

[53] Arber Zela, Thomas Elsken, Tonmoy Saikia, Yassine Marrakchi, Thomas Brox, and Frank Hutter. Understanding and robustifying differentiable architecture search. In *ICLR*, 2020. 1

[54] Arber Zela, Julien Siems, and Frank Hutter. Nas-bench-1shot1: Benchmarking and dissecting one-shot neural architecture search. In *ICLR*, 2020. 1, 2, 3

[55] Hongyi Zhang, Moustapha Cisse, Yann N. Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. In *ICLR*, 2018. 3

[56] Yiyang Zhao, Linnan Wang, Yuandong Tian, Rodrigo Fonseca, and Tian Guo. Few-shot neural architecture search. *arXiv*, 2020. 3

[57] Dongzhan Zhou, Xinchi Zhou, Wenwei Zhang, Chen Change Loy, Shuai Yi, Xuesen Zhang, and Wanli Ouyang. Econas: Finding proxies for economical neural architecture search. In *CVPR*, 2020. 2

[58] Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V. Le. Learning transferable architectures for scalable image recognition. In *CVPR*, 2018. 1, 2, 3