PICKING UP THE PIECES: SEPARATELY EVALUATING SUPERNET TRAINING AND ARCHITECTURE SELECTION

Anonymous authors
Paper under double-blind review

ABSTRACT

Differentiable Neural Architecture Search (NAS) has emerged as a simple and efficient method for the automated design of neural networks. Recent research has demonstrated improvements on various aspects on the original algorithm (DARTS), but comparative evaluation of these advances remains costly and difficult. We frame supernet NAS as a two-stage search, decoupling the training of the supernet from the extraction of a final design from the supernet. We propose a set of metrics which utilize benchmark data sets to evaluate each stage of the search process independently. We demonstrate two metrics measuring separately the quality of the supernet's shared weights and the quality of the learned sampling distribution, as well as corresponding statistics approximating the reliance of the second stage search on these components of the supernet. These metrics facilitate both more robust evaluation of NAS algorithms and provide practical method for designing complete NAS algorithms from separate supernet training and architecture selection techniques.

1 INTRODUCTION

Neural architecture search (NAS) describes the problem of automatically selecting an effective neural network architecture for given data. In their survey of work on NAS, Elsken et. al. establish three necessary components to achieve this goal: a search space to define the set possible architectures, a search strategy to traverse over potential solutions, and a performance estimation strategy to rate the quality of potential solutions considered by the search strategy (Elsken et al., 2019).

In recent years a variety of search spaces and benchmark problems have been proposed in NAS (Elsken et al., 2019). Exploring the space of search strategies has similarly generated vast interest, with approaches spanning evolutionary algorithms (Real et al., 2018), Bayesian optimization (Zela et al., 2018), reinforcement learning (Pham et al., 2018), and, most recently, direct gradient descent (Liu et al., 2019). Various methods for performance estimation have also been proposed, including weight inheritance (Real et al., 2017), predictive models (Liu et al., 2018), shortened training (Zoph et al., 2018), and weight-sharing (Pham et al., 2018). However, each of these components do not operate in a vacuum, and the interactions and relationships between them may dramatically affect the behavior and performance of the overall NAS algorithm. For example, performance estimates that extrapolate from previously explored models are biased by the search strategy's method for traversing the search space and differentially sampling different models.

This paper focuses on the class of algorithms which combine a weight-sharing performance estimation strategy and a gradient-based search strategy into the training of a single model. This approach trains a neural network representing a superposition of every architecture in the search space instead of training individual architectures and obtains performance estimates from this shared-weight model. This technique, which we will refer to as the supernet or supergraph method, was originally demonstrated by Pham et al. (2018), however the gradient-based training method contributed by DARTS (Liu et al., 2019) eliminated the need to train a second reinforcement learning model to select architectures, leading to the development of the series of NAS algorithms combining performance estimating and search in a single model considered here. Note that following Bender et al. (2018), the term "One-shot NAS" has been used to refer to supernet methods as they train a single model rather then iteratively training candidate architectures. However, as we will discuss iterative

search algorithms implemented on top the supernet, we adopt the supernet terminology used by Wang et al. (2021).

Differentiable supernet NAS algorithms have gained popularity due to their speed attained through weight-sharing – especially relative to employing the "gold standard" performance measure of training a neural network from multiple different random initializations and measuring the accuracy on a held-out set of validation data that often requires thousands of GPU hours for a single run (Zoph et al., 2018). They also benefit from and their ease of implementation within popular deep learning frameworks as a result of their reliance on gradient based optimization. These algorithms establish a special relationship between the search strategy and the performance estimation strategy, collapsing both components to the training of a single neural network. This presents a particular example of the phenomenon described above, where the shared weights are trained to maximize the performance of the previously explored architectures, and are then employed to estimate the performance of new potential architectures. The feedback loop between these two processes thus creates an opportunity to bias search towards familiar solutions, possibly at the expense of undervaluing architectures which have high train-from-scratch performance, but may not perform well with the weights learned on previously explored architectures. We might expect the degree to which this may be an issue to vary with the convexity of the search space, and the search strategy used to traverse that landscape.

In order to facilitate greater understanding of, and more robust comparisons between, differentiable supernet NAS algorithms, we present two statistics designed to evaluate separately the architecture search and performance estimation components of the supernet training process. We explain the value of these statistics by presenting a two stage framing of differentiable stochastic NAS, delinking the process of training the supernet from the process of selecting an architecture using the supernet. We then present a method for approximating the reliance of architecture selection methods on the performance estimation capabilities of the supernet versus the discovery of quality architectures through supernet training. Finally, we offer empirical results demonstrating the capabilities of using the presented statistics to design NAS algorithms through the composition of separate supernet training and architecture selection methods.

2 Related Work

2.1 Differentiable supernet NAS

DARTS was the first published differentiable NAS algorithm, utilizing a "continuous relaxation" method to parametrize the set of possible architectures, by placing trainable weights on each possible operation (Liu et al., 2019). The final model is selected from these trained architecture weights by simply keeping only the operations with the largest weight. This method is general enough to support the discovery of a broad range of neural network topologies through the use of a "zero" operation, representing the lack of an edge between nodes. Numerous works since the publication of DARTS have attempted to demonstrate simple improvements on DARTS to stabilize training, avoid the hard prune at the end of training, or improve scalability. PC-DARTS has demonstrated partial channel training, allowing the use of larger supernet models. Wang et al. (2021) addresses the final hard prune by removing the magnitude based prune and substituting a perturbation-based pruning algorithm. Chu et al. (2021) stabilizes the training process using an auxiliary skip connection, in contrast to prior work which relied on the Hessian eigenvalue regularizer.

In motivating SNAS, Xie et al. comment on the performance estimation aspect of DARTS, demonstrating the disparity in the validation accuracy obtained from the shared weight model of DARTS using the whole shared-weight model and the pruned final architecture (Xie et al., 2019). Stochastic differentiable NAS algorithms, like SNAS, offer one answer to this question of performance estimation in differentiable NAS through the use of sampling. These methods preserve the gradient-based approach of DARTS, while proposing a particular discrete architecture at each step. However, it is not necessarily the case that sampling during training means the shared weight model will actually learn to effectively estimate the performance of sampled architectures.

Bender et al. (2018) instead implemented a path-based form of dropout, randomly masking portions of the shared weight model during training and demonstrated a strong correlation between performance estimates from shortened training and the shared weight model, suggesting the two methods of performance estimation are comparable.

2.2 ZERO SHOT NAS

Recently, methods have been proposed to select an architecture without training even a single architecture model. Much of these methods are based on proxies, like EcoNAS (Zhou et al., 2020), while TE-NAS (Chen et al., 2021) recently demonstrated a zero-shot search based on measures emerging from deep learning theory. Abdelfattah et al. (2021) demonstrate a range of measures which have been used to compute saliency in other deep learning problems as possible measures to be useful in NAS.

2.3 SUPERNET NAS EVALUATION

Research on the evaluation of NAS methods has developed as a result of difficulties in comparing the performance NAS algorithms demonstrated in different architecture search spaces, as the selection of search space can have a greater effect on the final performance than the selection of NAS algorithm, which can be made clear through performing a random search (Li & Talwalkar, 2019).

Yu et al. (2020) initiated the critical inquiry into the use of weight-sharing for performance estimation by evaluating popular weight-sharing NAS algorithms in a reduced search space of 32 architectures. They demonstrated a lack of correlation in ranking between the performance estimates obtained from the shared-weight model and the trained from scratch test accuracy for and an improvement in performance of the final model achieved by not using weight-sharing. Further research evaluating the rankings of small samples (Yang et al., 2020) or small search spaces (Zhang et al., 2020) showed high variance in rankings across random seeds after training the shared weight model through random sampling.

NAS benchmark data sets have provided a significant advance in the available tools for NAS evaluation, with the first benchmark designed for the evaluation of supernet NAS algorithms provided by Dong & Yang (2020).

3 CASE STUDY: TWO MODIFICATIONS TO DARTS

Two recent papers demonstrated non-overlapping modifications to DARTS named DARTS- (Chu et al., 2021) and DARTS-PT (Wang et al., 2021). Each paper aimed to tackle the problem of unstable architecture weights in DARTS. DARTS- addressed this issue by adding an auxiliary skip connection to each mixed operation in DARTS, aiming to stabilize the training of architecture weights. DARTS-PT instead dismissed the architecture weights entirely, using a perturbation based method of selecting operations after training the supernet. The authors conclude that the perturbation technique may work better if DARTS is trained without architecture weights at all as they were able to demonstrate a smaller error using their method on a supernet trained with fixed architecture weights than on a supernet trained by DARTS. This result would suggest that the core innovation of DARTS, the continuous relaxation technique for approximating gradient updates to the architecture of the model, may not be of significant use. However, the generality of these conclusions are unclear from DARTS-PT alone. Given that the DARTS method of training architecture weights is notably unstable, do the results of DARTS-PT actually suggest that architecture weights are not reliable in general or merely that the original DARTS algorithm is so bad at training architecture weights that they are not worth using? The former would indeed suggest a major shortcoming of our understanding of supernet models, while the latter might imply a shortcoming in our ability to evaluate the holistic architecture search processes without understanding, and accounting for, the interactions between its shared-weight model training and architecture selection components.

	Max α	Perturb	Perturb w/ fix α
DARTS	45.7	11.89	6.20
DARTS-	6.75	6.56	7.22

Table 1: Comparison between original results of DARTS-PT (top line) showing improvement from not training architecture weights to results from including auxiliary skip connection that show improvement with architecture weights.

By implementing DARTS- within the provided codebase for DARTS-PT and replicating their tests, we observe a reversal of this result in Table 1. Once training of the architecture weights is stabilized via the auxiliary skip connection, training with architecture weights does provide an improvement to the perturbation based selection method. This result underscores the need for more tools for evaluating NAS algorithms in order to facilitate the development of general knowledge of the NAS process. Here, there was no evident way from the published results of either paper to predict the outcome of combining the two algorithms. Each paper demonstrated minor improvements when implemented with a different DARTS variant, but without explanation of how or why these algorithms are able to positively interact. While we might expect any two given modifications of an algorithm to have the potential to interact in any number of interesting and non-linear ways, in the next section we suggest that the interactions between the training of the shared-weight model and the selection of specific architectures from within this supernet may be able to provide insight into their combined effect if the two processes are evaluated separately before being considered in concert with one another.

4 TWO STAGE SEARCH

Much of the criticism of DARTS has focused specifically on the method by which the final architecture is extracted from the supernet. These issues with the "hard-prune" implemented in DARTS distract from evaluation of the process by which DARTS trains the supernet. To remedy this, we employ a two stage search framework for differentiable supernet-based NAS. We consider the training of the supernet as a "Stage-1 search algorithm" on its own. Although these algorithms do search the architecture space, their output is not a single final architecture but some form of information about the relative quality of architectures. We consider two possible forms this information can take: a weighting/sampling distribution over the architecture space (exemplified by α in DARTS) and a trained supernet model which able to provide performance estimates of individual candidate architectures. Note that the Stage-1 search algorithm need only provide one of these, as One-Shot NAS does not utilize any learned distribution over the architecture space, and the default Stage-2 search method for DARTS makes no use of the supernet model's shared weights.

As explained by Li & Talwalkar (2019), the original search method used for DARTS, actually consists of 3 stages: the supernet-based search, training the final model from several trials of supernet search partially from scratch, and further training of the best of these models. In contrast to this, we are focused solely on the process of supernet-based search. The second stage of our framework is not a proposal for an additional iterative step in search, but a decomposition of the existing process used to train a supernet and output a single architecture. We are simply denoting the architecture selection process inherent to all supernet-based NAS algorithms as "Stage-2 search." The purpose of this terminology choice is to draw attention to distinct components of existing NAS algorithms and investigate their relationship to each other. There are numerous ways that a final architecture can be obtained from any supernet training algorithm and, though some selection algorithms may be more suitable for particular training algorithms, no supernet training algorithm is necessarily or trivially tied to a particular architecture selection algorithm, or vice versa. We utilize the original names of

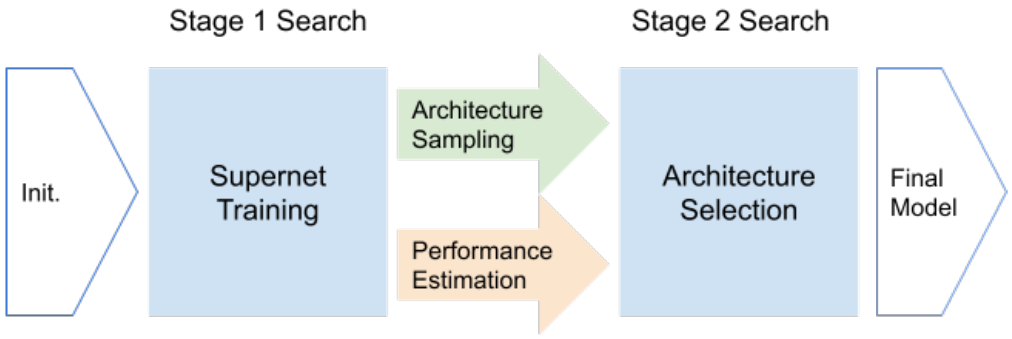

Figure 1: Diagram of our proposed 2-stage framework for differentiable NAS

the algorithms to denote Stage-1 search algorithms, such as DARTS and SNAS, while we name the Stage-2 search algorithms after the architecture selection criterion used.

Through this framing of two-stage supernet-based NAS, we are able to unify several disparate strands of NAS research. As the default Stage-2 search method of DARTS is notoriously unstable, techniques such as that of Wang et al. (2021) have been developed to bypass it. At the same time, zero-cost NAS methods, such as those of Abdelfattah et al. (2021) have provided advanced statistics for the rapid evaluation of sampled architectures. Positioning both of these techniques as Stage-2 search algorithms allows us to broaden the application of the former beyond DARTS and integrate the latter with supernet-based NAS techniques.

We evaluate both search stages and their combination using NAS-Bench-201.

4.1 Evaluating Stage-1 search

We consider the process of training a supernet to be Stage-1 of the architecture search. We focus specifically on a class of DARTS-like algorithms, that utilize gradient based techniques to learn a parameterization of the architecture space in the process of training the supernet. We wish to, as separately as possible, evaluate how these algorithms train their architecture and shared weights. As NAS-Bench-201's search space allows a single operation at each edge, the architecture weights, after a softmax, correspond exactly to a categorical distribution over operations. From the probability of each operation in a model, we can take their product to find the sampling probability mass assigned to each architecture. We utilize the benchmark data set to compute the probability of sampling architectures with any given test accuracy, allowing us to evaluate the architecture weights by directly observing the sampling probability mass they have assigned to the highest performing models in the search space. In order to evaluate the shared weights learned by the supernet training algorithms, we utilize them to record a validation accuracy for every architecture in the search space. We then rank every architecture according to its approximated performance and compare this ranking to the ground truth ranking obtained using the benchmark test accuracy. Although only one Stage-2 method we evaluate actually directly relies on validating individual architectures using the supernet, by obtaining performance estimates for each architecture in the search space and then attempting to rank them, we provide a proxy measure the amount of information about the relative quality of individual architectures within the supernet, which should also play a role in determining any other statistic computed over individual architectures using the supernet.

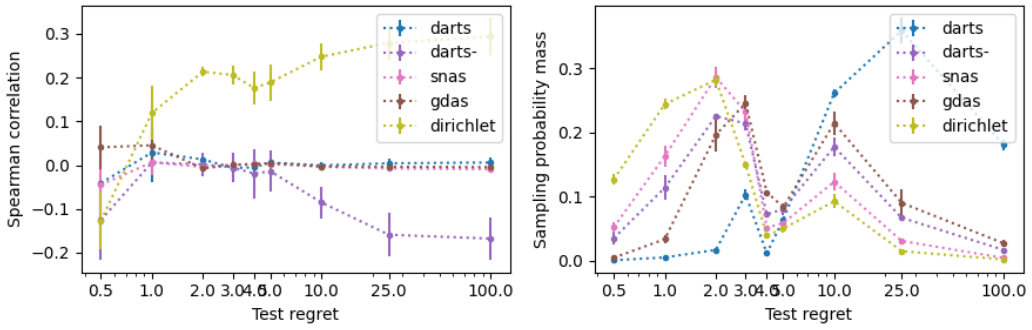

Figure 2: Left: Spearman correlation of shared-weight based validation accuracy and benchmark test accuracy across test regret threshold groups of the architecture space. Right: Sampling probability mass across test regret threshold groups of the probability space. The correlation is computed across all architectures below the threshold while probability mass is aggregated across architectures below the threshold and above the previous threshold.

In order to demonstrate how these statistics vary across the architecture space, we have aggregated the architecture space based on the decrease in performance compared to the top model in the search space (test regret). In order to depict sampling probabiblity mass, we have computed the total sampling probability mass between each test regret threshold. That is, each point corresponds to the

probability of sampling an architecture no worse preforming than the given value but no better than the previous threshold.

Figure 2 shows the Spearman correlation between the shared-weight-measured validation accuracy and the benchmark test accuracy for all architectures under the given threshold. The high variability in correlation displayed by the lowest test regret thresholds is the result of the small number of included architectures in the top thresholds. As the threshold increases and more architectures are added, the measured Spearman correlation stabilizes.

Stage-1 search algorithms we evaluated include DARTS, DARTS-, DrNAS, GDAS, and SNAS. DrNAS, GDAS, and SNAS are all sampling based NAS algorithms, with SNAS using a gumbel-softmax, GDAS using a gumbel-max, and DrNAS using a dirichlet distribution to sample an architecture at each step and backpropagate through the architecture weights.

In order to produce scalar statistics for the purpose of comparing Stage-1 search models, we compute the sampling probability assigned to the top 100 architectures in the search space, which we denote as α_1. We compute the Spearman correlation of the validation and benchmark test accuracies for every fifth of the top 500 architectures. As the top 95 architectures on CIFAR10 are all within 0.5 test regret, differentiating between them does not provide a useful measure of the capability of the supernet to distinguish good model from mediocre models.

4.2 Evaluating Stage-2 search

In order to evaluate the architecture selection process, we define a standardized input for the Stage-2 search algorithms. We wish to provide a sufficiently broad baseline such that it is suitable for evaluating any Stage-2 search algorithm which can be implemented on top of a supernet training algorithm. In order to satisfy this requirement, we must consider the supernet itself as the input to the architecture selection algorithms. Therefore, our baseline must consist of a standardized set of supernets, trained in a manner such that evaluating architecture selection methods on them facilitates estimation of their efficacy across a range of Stage-1 search methods.

Unfortunately, there is no readily apparent method for training architecture weights that could serve as a suitable baseline. As the method by which they train the architecture weights is the defining characteristic for several of the Stage-1 search algorithms we examine and the simplest method to train the architecture weights is already one of the algorithms (DARTS) it is not feasible to design a baseline training method which involves training the architecture weights without biasing the baseline towards evaluating compatibility with a specific Stage-1 algorithm.

As our baseline supernet training method can not involve training weights, this baseline will not support the evaluation of the default architecture selection method for DARTS, pruning by architecture weight magnitude. However, it is only exactly this class of architecture selection method, those which operate directly on the values of the architecture weights, rather than using them to parameterize a sampling distribution over the architecture space, that are unable to be evaluated on this baseline. As demonstrated by DARTS-PT, this baseline is not exclusive of Stage-2 search algorithms that operate directly on the operation level, only those that rely directly on the value of the architecture weight to do so.

The core of our baseline Stage-2 evaluation technique is given by the algorithm "Random Search with Parameter Sharing" (RSPS) proposed by Li & Talwalkar (2019) as a baseline method for evaluating the contribution of weight-sharing in NAS. Additionally, sampling architectures uniformly at random and training the supernet by adjusting the weights for a single architecture at a time presents the simplest method for training a supernet.

In order to simulate the benefit of architecture weights without actually training architecture weights, we utilize the relationship between the architecture weights and architecture sampling. As we are using a NAS benchmark to support this evaluation method, we have access to the ground truth performance of every architecture in the search space and can design an arbitrarily good distribution over the architecture space. As such, we augment our RSPS training method by replacing the uniform sampling with a sampling distribution where the probability of sampling a model is inversely proportional to its performance (so the sampling probability linearly decreases over the ranking by test accuracy). We also utilize this distribution for evaluation of Stage-2 search algorithms which sample architectures from the supernet.

In addition to these two trained supernets, we run each Stage-2 search algorithms on a randomly initialized model. We compute statistics corresponding to the reliance of the Stage-2 algorithm on the architecture discovery and performance estimation via supernet shared weights by taking the difference in performance on these baseline models. The measure corresponding to reliance on the supernet having learned effective architecture parameters, which we denote α_2 is computed as the difference between the performance of the biased sampling supernet and the uniform sampling supernet. The measure corresponding to reliance on obtaining information about the effectiveness of individual candidate architectures using the supernet's weights, which we denote β_2, is computed as the difference between performance on the uniform sampling supernet and the randomly initialized supernet.

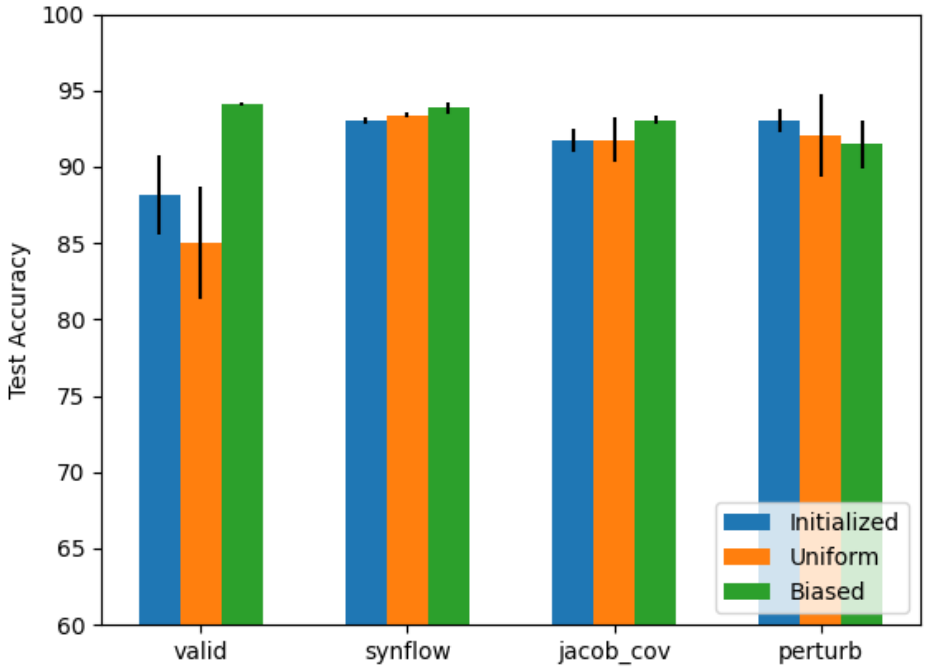

Figure 3: Comparison of Stage-2 search algorithm performance with given baseline models. "Initial" refers to untrained model, "Uniform" refers to a model trained by training a uniformly randomly sampled architecture at each batch, and "Biased" refers to a model trained by an architecture sampled from an arbitrarily good distribution at each batch.

We examine four techniques for Stage-2 search: searching based on the validation accuracy obtained from the shared weight model ("valid"); the top two measure reported in Abdelfattah et al. (2021), synflow and jacob_cov, and the perturbation based method demonstrated in Wang et al. (2021) ("perturb"). The first 3 methods are sampling based random searches: a sample of 100 architectures is drawn from the architecture distribution provided by the supernet and then the canditate architectures are ranked by either the validation accuracy, synflow, or jacobian covariance, outputting the highest ranked architecture. The perturbation based method instead operates directly on the supernet, iteratively masking out operations and computing validation accuracies to prune operations one by one until a final model is produced.

5 Putting it Together

Separate measures for evaluating Stage-1 and Stage-2 search facilitate estimation of the quality of combining Stage-1 and Stage-2 methods. To empirically test the utility of these measures, we combine the separate evaluation methods via a simple ranking technique to generate a unified ranking.

We compared this estimated ranking based on separate evaluation to the observed performance of each combination of search Stage-1 and search Stage-2 implemented on NAS-Bench-201.

We denote the ranking of Stage-1 search algorithms by sampling probability as $r_{\alpha 1}$ and the ranking by correlation of shared-weight validation accuracy with benchmark test accuracy as $r_{\beta 1}$. For Stage-2 models we compute the performance on the uniform baseline as $r_{\alpha 2}$ and the performance on the biased baseline as $r_{\beta 2}$. Then we compute a score for each combination of Stage-1 and Stage-2 algorithm as follows:

$$score = r_{\alpha 1}(|r_{\alpha 1} - r_{\alpha_2}| + 1) \times r_{\beta 1}(|r_{\beta 1} - r_{\beta 2}| + 1)$$

The scoring formula uses the difference between ranks to capture a mismatch in the capabilities of Stage-1 algorithms and those relied on by the Stage-2 search algorithms, adding 1 to avoid attributing undue weight to the Stage-1 and Stage-2 algorithms sharing a rank. We then scale this difference by the Stage-1 ranks as they are approximations of the supernet model's performance estimation and top architecture discovery capabilities, which should correlate with overall performance. As the Stage-2 ranks are computed using the difference in performance between pair of models rather than being a measure of model performance, we do not incorporate their magnitude. Given that NAS algorithms have been demonstrated which do not rely one of either a learned distribution over architectures or ever using the supernet to evaluate individual architectures, we use multiplication to combine the two separate measures so that a combination which succeeds at one of the two measures is scored above one which is mediocre at both.

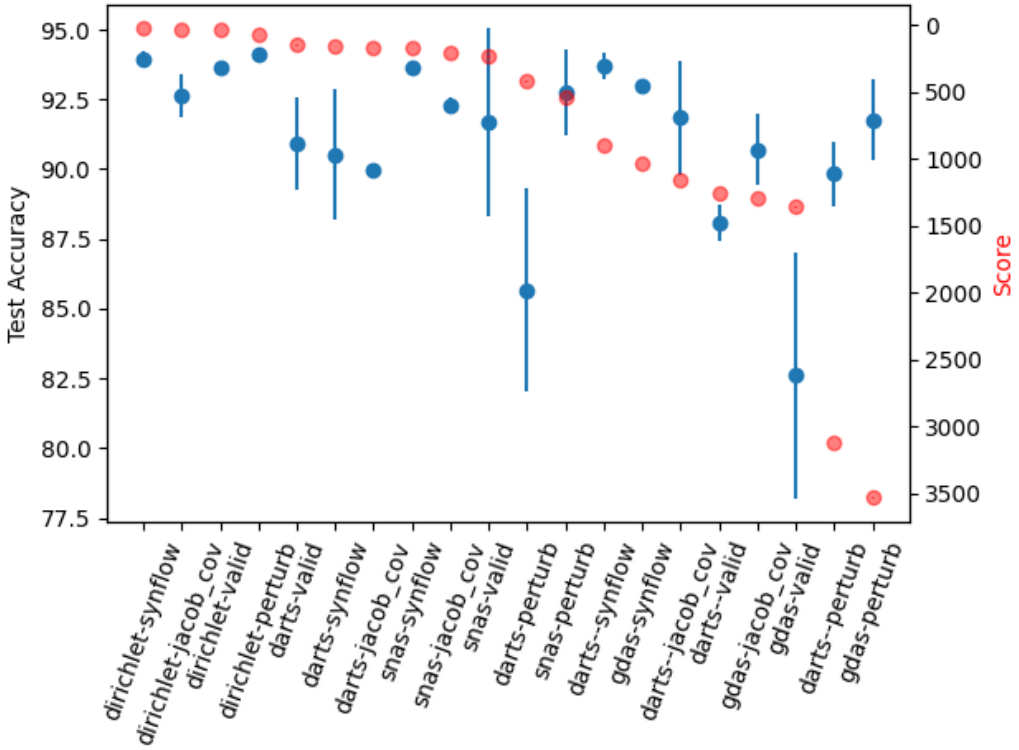

Figure 4: Performance of Stage-1 and Stage-2 search algorithm combinations, ordered by decreasing score, with benchmark test accuracy plotted in blue on the left axis and score plotted in red on the right axis.

After computing the scores, we then test the Stage-2 search algorithms on each Stage-1 algorithm, producing measurements of the benchmark test accuracy of the final architecture we are able to compare against the ranking implied by our scores, as shown in Fig. 5.

These final scores, with the accompanying ordering, suggest that these measures are a useful, but imperfect message for estimating the combined performance of a supernet training algorithm and architecture selection algorithm.

The purpose of this provided ranking is not necessarily to propose a specific method for ranking combinations of Stage-1 and Stage-2 algorithms but rather to provide an empirical demonstration of the usefulness of the proposed evaluation techniques. These evaluation techniques can be utilized by researchers who develop new Stage-1 or Stage-2 algorithms to provide additional information about how their algorithm functions and how it might interact with other NAS techniques that is not provided when only the final test accuracy is reported.

We also evaluate each Stage-1 and Stage-2 algorithm and their combinations on CIFAR100, with the result provided in the Appendix.

6 Conclusion

In order to facilitate integration of recent advances in NAS and well as more robust evaluation in NAS research, we have reframed existing differentiable supernet NAS algorithms as combinations of two separate search algorithms, the first being an implicit search via the process of training the supernet using architecture weights and the second being the process of selecting an architecture using the supernet. We propose two statistics approximating the quality of the weighting over the architecture space and the capacity of the supernet to provide information about the relative quality of individual canditate architectures, as well as two corresponding statistics approximating the reliance of architecture selection methods on each of these components of the supernet. We demonstrate an empirical evaluation of the proposed statistics by using them to compute a score for each combination of 5 Stage-1 search algorithms and 4 Stage-2 search algorithms and comparing the predicted ranking with the benchmark test accuracy obtained by each combination. Future work may be able to expand upon these methods, to develop measures suitable for NAS algorithms in which Stage-1 and Stage-2 search overlap in time through progressive pruning of the search space, measures which account for the stability of NAS algorithms across trials with multiple random seeds, or better evaluate Stage-2 methods that prune by the magnitude of architecture weights.

References

Mohamed S. Abdelfattah, Abhinav Mehrotra, Łukasz Dudziak, and Nicholas D. Lane. Zero-Cost Proxies for Lightweight NAS. In *International Conference on Learning Representations (ICLR)*, 2021.

Gabriel Bender, Pieter-Jan Kindermans, Barret Zoph, Vijay Vasudevan, and Quoc Le. Understanding and simplifying one-shot architecture search. In Jennifer Dy and Andreas Krause (eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 550–559, Stockholmsmässan, Stockholm Sweden, 10–15 Jul 2018. PMLR. URL `http://proceedings.mlr.press/v80/bender18a.html`.

Wuyang Chen, Xinyu Gong, and Zhangyang Wang. Neural architecture search on imagenet in four gpu hours: A theoretically inspired perspective. In *International Conference on Learning Representations*, 2021.

Xiangxiang Chu, Xiaoxing Wang, Bo Zhang, Shun Lu, Xiaolin Wei, and Junchi Yan. Darts-: Robustly stepping out of performance collapse without indicators. In *International Conference on Learning Representations (ICLR)*, 2021.

Xuanyi Dong and Yi Yang. Nas-bench-201: Extending the scope of reproducible neural architecture search. In *International Conference on Learning Representations*, 2020. URL `https://openreview.net/forum?id=HJxyZkBKDr`.

Thomas Elsken, Jan Hendrik Metzen, and Frank Hutter. Neural architecture search: A survey. *Journal of Machine Learning Research*, 20(55):1–21, 2019. URL `http://jmlr.org/papers/v20/18-598.html`.

Liam Li and Ameet Talwalkar. Random search and reproducibility for neural architecture search. In *Conference on Uncertainty in Artificial Intelligence*, July 2019.

Chenxi Liu, Barret Zoph, Maxim Neumann, Jonathon Shlens, Wei Hua, Li-Jia Li, Li Fei-Fei, Alan Yuille, Jonathan Huang, and Kevin Murphy. Progressive neural architecture search. In Vittorio Ferrari, Martial Hebert, Cristian Sminchisescu, and Yair Weiss (eds.), *Computer Vision – ECCV 2018*, pp. 19–35, Cham, 2018. Springer International Publishing. ISBN 978-3-030-01246-5.

Hanxiao Liu, Karen Simonyan, and Yiming Yang. DARTS: Differentiable architecture search. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=S1eYHoC5FX.

Hieu Pham, Melody Y. Guan, Barret Zoph, Quoc V. Le, and Jeff Dean. Efficient neural architecture search via parameter sharing. In *ICML*, 2018.

Esteban Real, Sherry Moore, Andrew Selle, Saurabh Saxena, Yutaka Leon Suematsu, Jie Tan, Quoc V. Le, and Alexey Kurakin. Large-scale evolution of image classifiers. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, pp. 2902–2911. JMLR.org, 2017.

Esteban Real, Alok Aggarwal, Yanping Huang, and Quoc V. Le. Regularized evolution for image classifier architecture search. In *AAAI*, 2018.

Ruochen Wang, Minhao Cheng, Xiangning Chen, Xiaocheng Tang, and Cho-Jui Hsieh. Rethinking architecture selection in differentiable nas. In *International Conference on Learning Representations (ICLR)*, 2021.

Sirui Xie, Hehui Zheng, Chunxiao Liu, and Liang Lin. SNAS: stochastic neural architecture search. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=rylqooRqK7.

Antoine Yang, Pedro M Esperanca, and Fabio Maria Carlucci. NAS evaluation is frustratingly hard. In *International Conference on Learning Representations*, 2020. URL https://arxiv.org/pdf/1912.12522.pdf.

Kaicheng Yu, Christian Sciuto, Martin Jaggi, Claudiu Musat, and Mathieu Salzmann. Evaluating the search phase of neural architecture search. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=H1loF2NFwr.

Arber Zela, Aaron Klein, Stefan Falkner, and Frank Hutter. Towards automated deep learning: Efficient joint neural architecture and hyperparameter search. In *International Conference on Machine Learning AutoML Workshop*, 2018. URL https://arxiv.org/pdf/1807.06906.pdf.

Yuge Zhang, Zejun Lin, Junyang Jiang, Quanlu Zhang, Yujing Wang, Hui Xue, Chen Zhang, and Yaming Yang. Deeper Insights into Weight Sharing in Neural Architecture Search. *arXiv e-prints*, art. arXiv:2001.01431, January 2020.

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 *Conferenc on Computer Vision and Pattern Recognition (CVPR*, 2020.

Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V. Le. Learning transferable architectures for scalable image recognition. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8697–8710, June 2018. doi: 10.1109/CVPR.2018.00907.

A APPENDIX

A.1 CIFAR-100 RESULTS

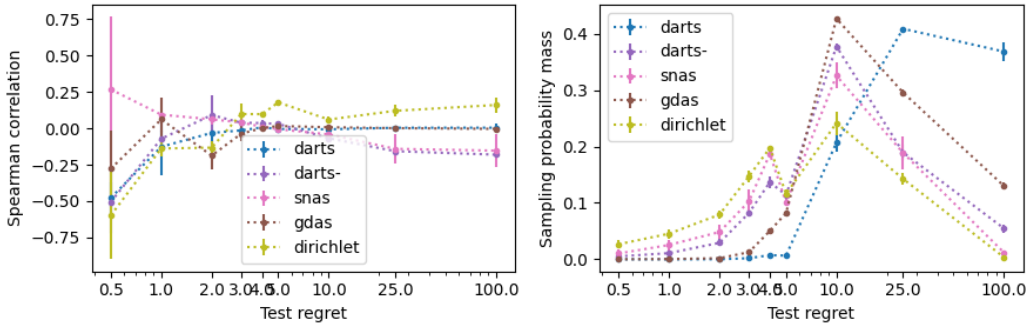

Figure 5: Left: Spearman correlation of shared-weight based validation accuracy and benchmark test accuracy across test regret threshold groups of the architecture space. Right: Sampling probability mass across test regret threshold groups of the probability space.

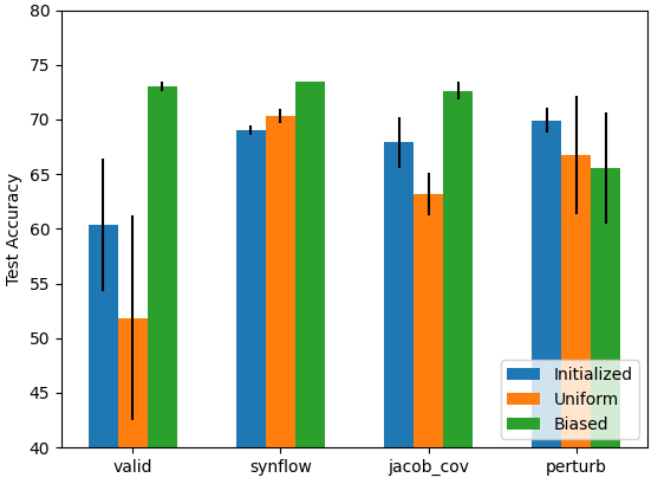

Figure 6: Comparison of Stage-2 search algorithm performance on CIFAR-100 with given baseline models. "Initial" refers to untrained model, "Uniform" refers to a model trained by training a uniformly randomly sampled architecture at each batch, and "Biased" refers to a model trained by an architecture sampled from an arbitrarily good distribution at each batch.

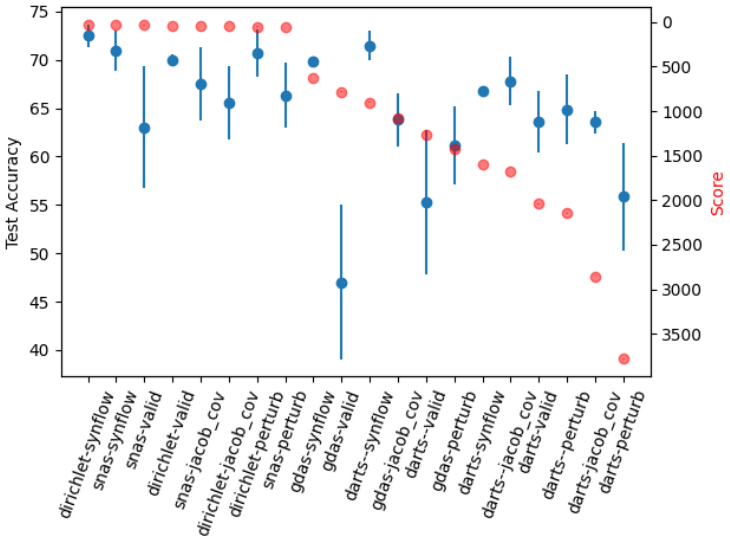

Figure 7: Performance of Stage-1 and Stage-2 search on CIFAR-100 algorithm combinations, ordered by score.