BNNAS++: Towards Unbiased Neural Architecture Search With Batch Normalization

YICHEN ZHU1 AND XIAOWEI FU2
1School of Mathematics and Statistics, University of New South Wales, Sydney, NSW 2052, Australia
2Sanda University, Shanghai 201209, China
Corresponding author: Xiaowei Fu (xwfu@sandau.edu.cn)

ABSTRACT Neural Architecture Search (NAS) achieves significant progress in many computer vision tasks, yet training and searching high-performance architectures over large search space are time-consuming. The NAS with BatchNorm (BNNAS) is an efficient NAS algorithm to speed up the training of supernet ten times over conventional schemes by fixing the convolutional layers and training the BatchNorm only. In this paper, we systematically examine the limitation of the BNNAS and present improved techniques. We observe that the BNNAS is prone to select convolution and fails to find plausible architectures with small FLOPs, especially on the search space with operations other than convolutions. This is because the number of learnable parameters on BatchNorm is imbalanced for different operations, i.e., BatchNorm is only attached to the convolutional layer. To fix the issue, we present the unbiased BNNAS for training. However, our empirical evidence shows that the performance indicator in BNNAS is ineffective on the unbiased BNNAS. To this end, we propose a novel performance indicator, which decomposes the model performance into model expressivity, trainability, and uncertainty, to preserve the rank consistency on the unbiased BNNAS. Our proposed NAS method, named as BNNAS++, is robust on various search spaces and can efficiently search high-performance architectures with small FLOPs. We validate the effectiveness of BNNAS++ on NAS-Bench-101, NAS-Bench-201, DARTS search space, and MobileNet search space, showing superior performance over existing methods.

INDEX TERMS Deep learning, automated machine learning, neural architecture search.

I. INTRODUCTION
Automatically finding the optimal neural architectures for a given task and dataset have been an active research field in recent years. The goal is to relieve human from tedious model design jobs and prevents numerous trial-and-error experiments. Pioneer works search architectures via reinforcement learning [1] or evolutionary algorithms [2], where they involve unbearable searching cost and resource consumption. Many NAS algorithms manage to use reduced computational settings, such as weight sharing NAS [3], [4], [5], [6], and cheap proxy [7]. But they still suffer the significant memory and computational cost.

To address the issue, recently neural architecture search with batch normalization (BNNAS) [8] was introduced to train only the BatchNorm layers while fix the convolutional layers for the weight-sharing supernet. In the search stage, the BNNAS compares the value of $\gamma$, the scale parameters in the BatchNorm, to indicate the model performance during the search stage. It speed up the supernet training up to ten times than conventional supernet training methods such as SPOS [6] and FairNAS [9], and achieve comparable results. In this paper, we systematically study the limitation of the BNNAS and present improved techniques for both training and searching. We observe that the original BNNAS is lean to select the convolutions and unable to find plausible architectures in the search space containing operations other than convolutional layers, such as pooling or identity layer. This is because only the convolutions are stacked with BatchNorm layers, which causes those architectures with more convolutions to have more learnable BatchNorm parameters than those with pooling or identity layers. Such imbalance...
in the number of learnable parameters causes the BNNAS to become bias towards convolutional layers. To this end, we present the unbiased BNNAS which maintains the same number of learnable BatchNorm parameters for all architecture; thus, these architectures can make a fair comparison during the search stage. However, we empirically find that the $\gamma$-indicator, the performance indicator that is previously proposed in BNNAS for architecture search, is ineffective on the unbiased BNNAS. Therefore, we introduce a novel performance indicator by combining model expressivity, trainability, and uncertainty measurements. Through extensive experiments, we verify that the composite indicator strongly correlates with the model test accuracy in the early training stage, and thus the supernet training time can be further reduced. We validate our BNNAS++ on NAS-Bench-101, NAS-Bench-201, DARTS search space, and MobileNet search space, showing promising performance and superior efficiency over existing methods.

### II. ANALYSIS OF NEURAL ARCHITECTURE SEARCH WITH BATCH NORMALIZATION

#### A. REVISIT THE BNNAS

First of all, we revisit the BNNAS approach. Given an input $x$, for any operations $OP(\cdot)$, it is common to map the parameters along the channel dimension to zero mean Gaussian distribution using batch-wise statistics $\mu$ and $\sigma$, and then rescale the parameters with $\gamma$ and $\beta$. Formally, we can describe the batch normalization as

$$ y = \gamma \frac{OP(x) - \mu}{\sqrt{\sigma + \epsilon}} + \beta $$

(1)

where $y$ is the output after BatchNorm, $\epsilon$ is an extremely small number to prevent zero on the denominator. The BNNAS limit the choice of $OP(\cdot) \in \text{Conv}(\cdot)$, nevertheless, a typical NAS search space have many types of operations $OP(\cdot) \in \{3 \times 3 \_\text{conv}, \text{avg} \_\text{pool}, \ldots, \text{identity}\}$. The BNNAS uses the $\gamma$ value as the indicator to search for the best architecture instead of validation accuracy. The conventional NAS methods train both $OP(\cdot)$ and BatchNorm together and then use the validation accuracy to rank the architectures. Instead, the BNNAS [8] propose to update the BatchNorm only and fix the parameters of $OP(\cdot)$. In the search stage, it uses the average $\gamma$ value over all cells to indicate model rank.

#### B. WHY BNNAS IS BIASED?

Figure 2 shows the scatter plot of trained model test accuracy versus the value of gamma in BatchNorm. We randomly sample 1,000 models from NAS-Bench-201 and train BatchNorm only with each model by one epoch on CIFAR10. We note that the original BNNAS trains the supernet for ten epochs in a weight-sharing setting, which produces a worse performance for each subnetwork than training a stand-alone model for
FIGURE 2. Analysis of gamma value of BatchNorm and test accuracy on CIFAR10. The size of the models that are measured by number of parameters (Left) and FLOPs (Right). Best view in colors. Conclusion: the BNNAS are biased toward large models.

TABLE 1. We count the number of various operations for the Top-10 architecture with training stand alone architectures in NAS-Bench-201 on CIFAR-10. We present the result of BNNAS and unbiased BNNAS.

| Method            | 1x1 Conv | 3x3 Conv | Avg Pool | Identity | Zero | Avg Params | Avg FLOPs | Avg Acc |
|-------------------|----------|----------|----------|----------|------|------------|-----------|---------|
| BNNAS             | 24       | 36       | 0        | 0        | 11   | 0.87       | 126       | 93.24   |
| Unbiased BNNAS    | 16       | 26       | 4        | 3        | 11   | 0.69       | 103       | 93.28   |

one epoch. Each color represents a range of the number of parameters.

We can observe that there is indeed a positive correlation between test accuracy and gamma value. However, the architectures with a high gamma value (over 85) mostly have a more significant number of parameters (we show the corresponding plot measured by FLOPs in Appendix). Meanwhile, the test accuracy of these architectures is mostly comparable or even worse than many smaller architectures with low gamma values. In Table 1 we count the number of operations in the architectures with the top ten highest gamma values in the BNNAS. We find out that the BNNAS favors convolutional layers: None of the architecture with high gamma values have operations other than 1 \times 1 and 3 \times 3 convolutions.

We argue that the BNNAS is biased towards convolution layers because the BatchNorm only attached to the convolutions. In other words, since only the BatchNorm is trained, the capacity of networks is largely determined by the learning process on the learnable parameters in BatchNorm. Even though we calculate the average value of gamma over all operation nodes, placing BatchNorm after convolutional layers only have a negative impact on searching good architectures with small sizes. In conclusion, the BNNAS is biased, cause unfairness on the search in supernet.

C. UNBIASED BNNAS

It is necessary to fix the unfairness in the BNNAS, thus we propose the unbiased BNNAS. The method is simple: we attach a BatchNorm layer to every operation, including pooling layer, identity layer, etc. As such, a fair evaluation for each architecture can be done when the training is finished. This is extremely important for weight sharing NAS [9], [10], [11], [12]. The top part in Figure 1 gives a comparison of the BNNAS and the unbiased BNNAS in the training stage. We also show the number of operations on the top ten architectures search by the unbiased BNNAS in Table 1. Evidently, our approach can find networks consist of diverse operations. While obtaining slightly better accuracy than the BNNAS, our approach consistently finds smaller architecture (measure by both FLOPs and number of parameters.).

D. THE INEFFECTIVENESS OF $\gamma$-INDICATOR

The most straightforward way to search architectures is to reuse the $\gamma$-indicator for unbiased BNNAS. However, our empirical study shows that the correlation between test accuracy and gamma value no longer exists after the training becomes fair for various network architectures. The Left Figure 3 shows the results of stand-alone training networks on CIFAR10 with train BatchNorm only for one epoch and two hundred epochs. There are 1,000 architectures randomly sampled from NAS-Bench-101. We can observe when the networks are trained for only one epoch, the correlation between test accuracy and gamma value is completely random. The architectures with high gamma value can have an accuracy range from 30% to 93%. When we train the networks for 200 epochs, the situation become worse. In the right Figure 3, the architecture that obtain the highest gamma value over 200 achieves around 60% test accuracy. As we attach BatchNorm after pooling layers or identity layers, the
gamma value of network architectures with those operations can also reach a very high value, which makes the $\gamma$-indicator no longer be an effective performance indicator for unbiased BNNAS.

III. BNNAS++: COMPOSITE INDICATOR FOR UNBIASED BNNAS

In the previous section, we have shown that the necessity of using unbiased BNNAS and how $\gamma$-indicator fails on it. In this section, we introduce a surrogate performance indicator for unbiased BNNAS. Our goal is to design an indicator that can be 1) effectively predicted the rank of the architectures at early training stage of the unbiased BNNAS, and 2) efficiently calculated during search stage. We disentangle the model generalization into model expressivity, trainability, and uncertainty. We provide detailed description of each individual indicator and empirically demonstrate their correlation with the model performance. We also conduct ablation on essential components that might impact the results of our method, for instance, the batch size and initialization methods in Appendix.

Expressivity: The ability of deep networks that can compactly express highly complex functions over input space is viewed as one of the most critical factors to their success. The expressivity of deep networks has been widely studied theoretically [13], [14] and has been used on NAS to score networks with random features [15], [16]. However, both [16] and [15] count the number of linear regions of randomly initialized networks, limiting their search space on networks with ReLU activation [17]. Whereas current SOTA backbones highly rely on non-linear activation other than ReLU, such as Swish [18], [19].

This paper discusses the expressivity of random features for various layer choices, such as convolutional layers, pooling layers, identity, etc., with trainable batch normalization. The batch normalization can be considered as a reparameterization trick on the parameters space. Learning to scale and shift the initialized parameters does not increase the expressivity of the random feature itself. Therefore, we refer to the validation accuracy as the expressivity of the deep networks. The original BNNAS fails to measure the expressivity of random features due to unfair supernet training. In our unbiased BNNAS, since the number of learnable BatchNorm parameters are the same for each operation, we can adopt expressivity as a performance indicator.

Trainability: Besides the expressivity, whether a deep network can achieve good performance is determined by how effectively the optimizer can optimize it. Recent work [20] shows the effect of BatchNorm on the Lipschitzness of the loss, which plays a crucial role in optimization by controlling the amount by which the loss can change when taking a training step [21], [22]. Their empirical and theoretical results indicate that the BatchNorm smooth the loss landscape, and the re-parametrization in BatchNorm makes the gradient of the loss more Lipschitz.

Specifically, let us denote the loss of a network with BatchNorm as $L_{BN}$ and the loss of the same network without BatchNorm as $L_{\text{BN}}$. Given the activation $\hat{y}_j$ and gradient $\nabla_{\hat{y}} \hat{L}$, we define the Lipschitzness of the loss as $||\nabla_{\hat{y}} L||$, [20] prove that we can measure the effect of BatchNorm on the Lipschitz by

$$||\nabla_{\hat{y}} \hat{L}||^2 \leq \frac{\gamma^2}{\sigma_j^2} (||\nabla_{\hat{y}} L||^2 - \frac{1}{m} \langle 1, \nabla_{\hat{y}} L \rangle^2 - \frac{1}{\sqrt{m}} \langle \nabla_{\hat{y}} L, \hat{y} \rangle^2)$$

As a result, the scale term $\gamma$ in the inequality can be used to measure the flatness of the loss landscape, which we define as the trainability score in our context.

Uncertainty: Uncertainty is a measurement of knowing what the model does not know. Deep neural networks are known to be overconfident in their predictions. Instead of point estimation, the model can offer confidence bounds for each decision from the probabilistic view to avoid overconfidence. This is useful to solve core issues of deep networks such as poor calibration and data inefficiency [23], [24]. There are rarely studies that discuss the relationship between uncertainty and model generalization. In this work, we obtain practical uncertainty for each network architecture via trainable batch normalization [25]. We discuss the correlation of uncertainty and performance, and we use it as one of the performance indicators to find plausible architectures.

Specifically, given a network architecture $X$, we are interested in approximating inference in Bayesian modeling.

![Train BatchNorm for 1 Epoch](image1)
![Train BatchNorm for 200 Epochs](image2)

**FIGURE 3.** The gamma value of BatchNorm versus test accuracy train with 1 (left) and 200 (right) epoch. Conclusion: the $\gamma$-Indicator is ineffective on the unbiased BNNAS.
A common approach is to learn a parameterized approximating distribution $q_\theta(\omega)$ that minimizes $KL(q_\theta(\omega) \| p(\omega|D))$, where $D$ is the training set, $\omega$ is the model parameters, $p(\omega|D)$ is the probabilistic model, the Kullback-Leibler divergence is of the true posterior with respect to its approximation.

For models that are built upon batch normalization, we are able to obtain approximate posterior by modeling $\mu_B$ and $\sigma_B$ over a mini-batch $B$ as stochastic variables. Following [25], we use approximate posterior to express an approximate predictive distribution $p(y|x,D) = \int f_\omega(x,y)q_\theta(\omega)d\omega$, and we can obtain the covariance of the predictive distribution empirically by

$$
\text{Cov}_{p^*}[y] \approx \tau^{-1}I + \frac{1}{T} \sum_{t=1}^{T} f(\hat{\mu}_i, \hat{\sigma}_i)(x)^T f(\hat{\mu}_i, \hat{\sigma}_i)(x) - E_{p^*}[y]^T E_{p^*}[y]
$$

The calculation of variance is generic in practice. We simply record the value of variance for each iteration during evaluation and then average the total variance to obtain the final uncertainty score.

**A. EMPIRICAL STUDY ON CORRELATION ANALYSIS**

We verify each indicator on NAS-Bench-201 with CIFAR10, CIFAR100, and ImageNet16-120. We randomly sample 2,300 architectures and plot the scatter plot to show the correlation between each indicator and test accuracy on trained networks. The test accuracy is directly taken from [26]. All models are trained by BatchNorm with the same default settings in [26] for one epoch on the target dataset. We train the models that only update the BatchNorm layers for one epoch because we want to check that our proposed indicator is effective in the early training stage of the networks. The results are shown in Figure 4. We provide Kendall Tau correlation [27] metrics for each indicator on every dataset, and all p-value for the tau correlation is less than 0.005 to ensure the results are statistically significant.

We observe that 1) a negative correlation between initial expressivity and test accuracy, 2) a positive correlation...
between trainability and test accuracy, and 3) a negative correlation between uncertainty and test accuracy. These results indicate that, for networks that are fixed with convolutional layers and trained with BatchNorm only, at the initial stage, a better architecture in the search space is 1) perform bad, 2) easy to train, 3) certain about its predictions. It is plausible for the correlation on the trainability terms. A network architecture with good trainability is easier to converge to a wide optima [28] with better generalization. It is also not surprising that deep networks with more confidence (less uncertain) on the prediction tend to generalize better, which is similar to the studies on human investors [29], [30], and athletes [31].

Nevertheless, it is counter-intuitive that good architectures tend to perform worse at the early stage of training BatchNorm only. A possible explanation is that architectures trained with BatchNorm only tend to over-fit on the training data first, which is also observed by [32] and then generalize on the test data. The high expressivity networks are more likely to over-fit initially due to the tendency to make the feature representation become orthogonal [33], and will be more generalizable than low expressive networks if more training time is conducted. We provide more evidence on this topic in Appendix.

**B. COMPOSITE PERFORMANCE INDICATOR**

We present and verify the strong correlation between the network’s test accuracy and its expressivity of random features, trainability, and uncertainty, respectively. The key to build an effective NAS algorithm is how to combine these indicators. As shown in the Figure 4, the numerical value for each indicator varies. The naive approach would be to normalize each indicator’s value, then add them into a single score. However, such a way requires us to know the estimated range of the value prior to we perform the search, which is impossible for most cases. Therefore, we rank the architecture according to the value for each indicator instead. How to combine the rank results on different indicators can be treated as an ensemble learning problem. In practice, we give equivalent importance to each indicator and use the average rank across three indicators as the final rank for simplicity.

**IV. EXPERIMENTS**

In this section, we introduce the experiments on NAS-Bench-101, NAS-Bench-201, DARTS search space, and MobileNet search space. All search cost are evaluated on Nvidia GTX 1080-Ti. More detailed experimental settings and search space description can be found in Appendix.

**A. NAS-BENCHMARK EXPERIMENTS**

1) **NAS-BENCH-101**

[34] is the first reproductive benchmark for NAS, and we compare our BNNAS++ with random search [34], REA [2], AREA [15], and RLNAS [35]. As shown in Table 4, our
Algorithm 1: BNNAS++ With Evolution Search

Define the random architecture sampler
RandArchSampler(), and batch_size = 256

ep_rank, t_rank, uc_rank = list(), list(), list()

for $i = 1, \ldots, T$ do

Update TopK architectures as parents according to indicators.
Mutate TopK network arch = RandArchSampler()
Get data $x$ and target $y$ of randomly sampled mini-batch.

ep_rank.append(accuracy(arch($x$), $y$))
t_rank.append(arch.sum_bn_grad)
uc_rank.append(arch.var)

end

Find the best_arch with highest average rank over ep_rank, t_rank, and uc_rank.
return best_arch.

TABLE 3: The Kendall Tau correlation of the composite indicator vs. test accuracy on NAS-Bench-201 across three datasets.

| Indicator | CIFAR-10 | CIFAR-100 | ImageNet-16 |
|-----------|-----------|-----------|-------------|
| Expressed | -0.357    | -0.463    | -0.419      |
| Trainability | 0.431    | 0.398     | 0.364       |
| Uncertainty | -0.388    | -0.447    | -0.358      |
| Combined  | 0.563     | 0.648     | 0.551       |

BNNAS++ outperforms the BNNAS counterpart by a large margin, on an average accuracy of 94.16% versus 91.04%. Our approach also achieves slightly higher accuracy than other methods with much less search cost, for instance, 6.8% of RLNAS and 5.8% of non-weight sharing REA.

2) NAS-BENCH-201

The NAS-Bench-201 is proposed by [26], which benchmarks a number of NAS algorithms. It contains 15,625 candidate architectures and provides the accuracy for each architecture. We compare our proposed methods with non-weight sharing NAS (Random Search, REA [2], REINFORCE [36], BOHB [37]), weight-sharing NAS (Random Search [38], DARTS [3], GDAS [39], SETN [40], ENAS [4], FairNAS [9], BNNAS [8]). We report the experimental results of BNNAS++ in Table 2. Our results indicate the BNNAS++ is extremely fast: over 20 to 30 times faster than the weight-sharing approach, while we obtain comparable or slightly better performance than these methods. Compared to the BNNAS, our approach further save 70% of search time and achieve better performance on all three datasets.

TABLE 4: The mean±std. accuracy with search cost on NAS-Bench-101.

| Method    | Accuracy | Search Cost |
|-----------|----------|-------------|
| Random    | 90.38±5.51| N/A         |
| REA       | 93.87±0.22| 12000       |
| AREA      | 93.91±0.29| 12000       |
| RLNAS     | 93.78±0.14| 10372       |
| BNNAS     | 91.04±0.32| 979         |
| BNNAS++   | 94.16±0.10| 701         |

TABLE 5: The Top-1 accuracy on ImageNet with MobileNet search space and DARTS search space.

| Method                  | Acc (%) | FLOPs (M) | Search Cost (GPU days) |
|-------------------------|---------|-----------|------------------------|
| Search Space: MobileNet |         |           |                        |
| ProxylessNAS            | 75.1    | 465       | 8.3                    |
| FBNet                   | 74.9    | 375       | 9                      |
| AngleNAS                | 75.97   | 472       | 10                     |
| FairNAS                 | 74.07   | 325       | 16                     |
| SPOS                    | 75.73   | 470       | 12                     |
| BNNAS (FairNAS)         | 74.12   | 326       | 0.8                    |
| BNNAS++ (FairNAS)       | 74.28   | 325       | 0.5                    |
| BNNAS (SPOS)            | 75.67   | 470       | 1.2                    |
| BNNAS++ (SPOS)          | 75.81   | 465       | 0.8                    |
| BNNAS (SPOS) + SE       | 76.78   | 473       | 0.8                    |
| BNNAS++ (SPOS) + SE     | 76.83   | 327       | 0.5                    |
| Search Space: DARTS     |         |           |                        |
| DARTS                   | 73.3    | 574       | 4                      |
| GDAS                    | 74.4    | 590       | 0.3                    |
| PC-DARTS                | 75.8    | 597       | 3.8                    |
| Fair-DARTS              | 75.1    | 541       | 0.4                    |
| RLNAS                   | 75.9    | 597       | 8                      |
| BNNAS (SPOS)            | 74.7    | 598       | 0.5                    |
| BNNAS++ (SPOS)          | 75.7    | 534       | 0.33                   |

B. ImageNet EXPERIMENTS

We conduct experiments ImageNet [41] with two commonly used search space: MobileNet search space and DARTS search space, to compare BNNAS++ with state-of-the-art NAS.

1) MobileNet SEARCH SPACE

MobileNet-like Search Space consists of 21 to-be-searched layers, each layer leverages the MobileNetV2 inverted bottleneck [42] and optional with SE module [43]. At each layer, the search method can choose operation between convolutional layers with kernel size {3, 5, 7}, expansion ratio {3, 6} and identity layers. Note that the original BNNAS does not have the identity layer option. We compare with several weight-sharing NAS that use the same search space setting [5], [6], [8], [9], [44]. Specifically, we adopt the training strategy of SPOS and FairNAS. The supernet is trained via the unbiased BNNAS for five epochs and ten epochs. Then, we use our proposed indicator to search the architecture. As shown in Table 5, with the same search space, our searched results achieve 74.28% with 325 MFLOPs and 75.81% with 465 MFLOPs, slightly outperform SPOS and FairNAS counterpart, whereas the search time is only 24 times and 20 times less the of the original method, respectively. Furthermore, there is a clear advantage of BNNAS++ over the BNNAS, which outperforms the BNNAS by 0.16% and 0.14% on accuracy with 1.6 times and 1.5 times less search cost. It indicates that even on the search space that is mainly composed of convolutions, our method is still superior to the BNNAS. By searching the SE module, we obtain...
architecture with 1.05% higher accuracy than the BNNAS with only 69% of the FLOPs. Our unbiased training strategy and performance indicator help us find small architectures with plausible accuracy.

2) DARTS SEARCH SPACE

We compare our algorithms with various NAS algorithms [3], [9], [12], [35], [39] that performed on the same DARTS search space. The bottom of Table 5 provides the experimental results on ImageNet. We can observe that the BNNAS does not achieve satisfactory results, as the test accuracy is relatively lower than state-of-the-art methods, yet its FLOPs are similar or higher. On the other hand, the BNNAS++ obtain 1.0% higher accuracy and 64 fewer FLOPs than the BNNAS. It demonstrates that our approach is robust on search space with various operations. This further backs up our initial assumption that the biased BNNAS hurts search performance.

VI. CONCLUSION

In this work, we examine the limitation of the BNNAS, an efficient NAS algorithm. We observe that the BNNAS brings unfairness into the supernet training, and the search criterion is ineffective if the unfairness issue is fixed. To this end, we propose the unbiased BNNAS and a novel composite performance indicator to search for good architecture efficiently, effectively and robustly. Our proposed method BNNAS++ is evaluated on NAS-Bench-101, NAS-Bench-201, DARTS search space, and MobileNet search space, showing superior search efficiency and effectiveness. These results indicate that our approach makes successful usage of BatchNorm in NAS. Overall, we believe that our approach provides an interesting and practical new perspective on BatchNorm in NAS.

REFERENCES

[1] B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le, “Learning transferable architectures for scalable image recognition,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit., Jun. 2018, pp. 8697–8710.
[2] E. Real, A. Aggarwal, Y. Huang, and Q. V. Le, “Regularized evolution for image classifier architecture search,” in Proc. AAAI Conf. Artif. Intell., vol. 33, 2019, pp. 4780–4789.
[3] H. Liu, K. Simonany, and Y. Yang, “DARTS: Differentiable architecture search,” in Proc. Int. Conf. Learn. Represent., 2018, pp. 1–13.
[4] H. Pham, M. Guan, B. Zoph, Q. Le, and J. Dean, “Efficient neural architecture search via parameters sharing,” in Proc. Int. Conf. Mach. Learn., 2018, pp. 4095–4104.
[5] H. Cai, L. Zhu, and S. Han, “ProxylessNAS: Direct neural architecture search on target task and hardware,” 2018, arXiv:1812.00332.
[6] H. Cai, L. Zhu, and S. Han, “ProxylessNAS: Direct neural architecture search on target task and hardware,” 2018, arXiv:1812.00332.
[7] H. Cai, L. Zhu, and S. Han, “ProxylessNAS: Direct neural architecture search on target task and hardware,” 2018, arXiv:1812.00332.
[8] B. Chen, P. Li, B. Li, C. Lin, C. Li, M. Sun, J. Yan, and W. Ouyang, “BN-NAS: Neural architecture search with batch normalization,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2020, pp. 11396–11404.
[9] B. Chen, P. Li, B. Li, C. Lin, C. Li, M. Sun, J. Yan, and W. Ouyang, “BN-NAS: Neural architecture search with batch normalization,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2020, pp. 11396–11404.
[10] X. Chu, T. Zou, B. Zhang, and J. Li, “FairDarts: Eliminating unfair advantages in differentiable architecture search,” in Proc. Eur. Conf. Comput. Vis. Springer, 2020, pp. 465–480.
[11] X. Chu, T. Zou, B. Zhang, and J. Li, “FairDarts: Eliminating unfair advantages in differentiable architecture search,” in Proc. Eur. Conf. Comput. Vis. Springer, 2020, pp. 465–480.
[12] X. Chu, T. Zou, B. Zhang, and J. Li, “FairDarts: Eliminating unfair advantages in differentiable architecture search,” in Proc. Eur. Conf. Comput. Vis. Springer, 2020, pp. 465–480.
[13] B. Poole, S. Lahiri, M. Raghu, J. Sohl-Dickstein, and S. Ganguli, “Exponential expressivity in deep neural networks through transient chaos,” in Proc. Adv. Neural Inf. Process. Syst., vol. 29, 2016, pp. 3360–3368.
[14] M. Raghu, B. Poole, J. Kleinberg, S. Ganguli, and J. Sohl-Dickstein, “On the expressive power of deep neural networks,” in Proc. Int. Conf. Mach. Learn., 2017, pp. 2847–2854.
[15] J. Meller, J. Turner, A. Storkey, and E. J. Crowley, “Neural architecture search without training,” in Proc. Int. Conf. Mach. Learn., 2021, pp. 7588–7598.
[16] W. Chen, X. Gong, and Z. Wang, “Neural architecture search on ImageNet in four GPU hours: A theoretically inspired perspective,” in Proc. Int. Conf. Learn. Represent., 2021, pp. 1–15.
[17] G. Montúfar, R. Pascanu, K. Cho, and Y. Bengio, “On the number of linear regions of deep neural networks,” 2014, arXiv:1402.1869.
[18] A. Howard, M. Sandler, B. Chen, W. Wang, L.-C. Chen, M. Tan, G. Chu, V. Vasudevan, Y. Zhu, R. Pang, H. Adam, and Q. Le, “Searching for MobileNetV3,” in Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV), Oct. 2019, pp. 1314–1324.

[19] M. Tan and Q. Le, “EfficientNet: Rethinking model scaling for convolutional neural networks,” in Proc. Int. Conf. Mach. Learn., 2019, pp. 6105–6114.

[20] S. Satturkar, D. Tsipras, A. Ilyas, and A. Madry, “How does batch normalization help optimization?” in Proc. 32nd Int. Conf. Neural Inf. Process. Syst., 2018, pp. 2488–2498.

[21] S. Boyd, S. P. Boyd, and L. Vandenberghe, Convex Optimization, Cambridge, U.K.: Cambridge Univ. Press, 2004.

[22] Y. Nesterov, Introductory Lectures on Convex Optimization: A Basic Course, vol. 87. Springer, 2003.

[23] Y. Gal and Z. Ghahramani, “Dropout as a Bayesian approximation: Representing model uncertainty in deep learning,” in Proc. Int. Conf. Mach. Learn., 2016, pp. 1050–1059.

[24] A. Kendall and Y. Gal, “What uncertainties do we need in Bayesian deep learning for computer vision?” 2017, arXiv:1703.04977.

[25] M. Teye, H. Azizpour, and K. Smith, “Bayesian uncertainty estimation for batch normalized deep networks,” in Proc. Int. Conf. Mach. Learn., 2018, pp. 4907–4916.

[26] N. Wang, Y. Gao, H. Chen, P. Wang, Z. Tian, C. Shen, and Y. Zhang, “Auto-FPN: Automatic network architecture adaptation for object detection beyond classification,” in Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV), Oct. 2019, pp. 6649–6658.

[27] F. Bach, H. Abdi, “The Kendall rank correlation coefficient,” in Encyclopedia of Measurement and Statistics. Thousand Oaks, CA, USA: Sage, 2007, pp. 508–510.

[28] H. Li, Z. Xu, G. Taylor, C. Studer, and T. Goldstein, “Visualizing the landscape of neural nets,” 2017, arXiv:1712.09913.

[29] S. Gervais and T. Odean, “Learning to be overconfident,” Rev. Financial Stud., vol. 14, no. 1, pp. 1–27, Jan. 2001.

[30] J. B. D. Long, A. Shleifer, L. H. Summers, and R. J. Waldmann, “The survival of noise traders in financial markets,” J. Bus., pp. 1–19, 1991.

[31] D. L. Feltz, “Self-confidence and sports performance,” Tech. Rep., 2007.

[32] J. Frankle, D. J. Schwab, and A. S. Morcos, “Training batchnorm and only batchnorm: On the expressive power of random features in CNNs,” in Proc. Int. Conf. Learn. Represent., 2021, pp. 1–28.

[33] H. Daneshmand, A. Joudaki, and F. Bach, “Batch normalization orthogonalizes representations in deep random networks,” in Proc. Int. Conf. Mach. Learn., 2021, arXiv:2106.03970.

[34] C. Ying, A. Klein, E. Christiansen, E. Real, K. Murphy, and F. Hutter, “NAS-Bench-101: Towards reproducible neural architecture search,” in Proc. Int. Conf. Mach. Learn., 2019, pp. 7105–7114.

[35] X. Zhang, P. Hou, X. Zhang, and J. Sun, “Neural architecture search with random labels,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2021, pp. 10907–10916.

[36] N. Wang, Y. Gao, H. Chen, P. Wang, Z. Tian, C. Shen, and Y. Zhang, “NAS-FCONS: Fast neural architecture search for object detection,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2020, pp. 11943–11951.

[37] T. Elsenk, B. Staffler, A. Zela, J. H. Metzen, and F. Hutter, “Bag of tricks for neural architecture search,” Structure, vol. 27, no. 48, p. 37.

[38] Y. Lei Ba, J. R. Kiros, and G. E. Hinton, “Layer normalization,” 2016, arXiv:1607.06450.

[39] Y. Wu and K. He, “Group normalization,” in Proc. Eur. Conf. Comput. Vis. (ECCV), 2018, pp. 3–19.

[40] J. Lei Ba, J. R. Kiros, and G. E. Hinton, “Layer normalization,” 2016, arXiv:1607.06450.

[41] S. Ioffe and C. Szegedy, “Batch normalization: Accelerating deep network training by reducing internal covariate shift,” in Proc. Int. Conf. Mach. Learn., 2015, pp. 448–456.

[42] Y. Wu and J. Johnson, “Rethinking ‘batch’ in BatchNorm,” 2021, arXiv:2105.07576.

[43] H. Daneshmand, M. J. Kohler, F. R. Bach, T. Hofmann, and A. Lucchi, “Batch normalization provably avoids ranks collapse for randomly initialized deep networks,” in Proc. NeurIPS, 2020, pp. 18387–18398.

[44] E. Hoffer, I. Hubara, and D. Soudry, “Train longer, generalize better: Closing the generalization gap in large batch training of neural networks,” 2017, arXiv:1705.07841.

[45] G. Bender, P.-J. Kindermans, B. Zoph, V. Vasudevan, and Q. Le, “Understanding and simplifying one-shot architecture search,” in Proc. Int. Conf. Mach. Learn., 2018, pp. 550–559.

[46] H. Xu, L. Yao, Z. Li, X. Liang, and W. Zhang, “Auto-FPN: Automatic network architecture adaptation for object detection beyond classification,” in Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV), Oct. 2019, pp. 6649–6658.

[47] Y. Chen, T. Yang, X. Zhang, G. Meng, X. Xiao, and J. Sun, “DetNAS: Backbone search for object detection,” in Proc. Adv. Neural Inf. Process. Syst., vol. 32, 2019, pp. 6642–6652.

[48] N. Wang, Y. Gao, H. Chen, P. Wang, Z. Tian, C. Shen, and Y. Zhang, “NAS-FCONS: Fast neural architecture search for object detection,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2020, pp. 11943–11951.

[49] T. Elsenk, B. Staffler, A. Zela, J. H. Metzen, and F. Hutter, “Bag of tricks for neural architecture search,” Structure, vol. 27, no. 48, p. 37.