Unifying and Boosting Gradient-Based Training-Free Neural Architecture Search

Anonymous Author(s)
Affiliation
Address
email

Abstract

Neural architecture search (NAS) has gained immense popularity owing to its ability to automate neural architecture design. A number of training-free metrics are recently proposed to realize NAS without training, hence making NAS more scalable. Despite their competitive empirical performances, a unified theoretical understanding of these training-free metrics is lacking. As a consequence, *(a)* the relationships among these metrics are unclear, *(b)* there is no theoretical interpretation for their empirical performances, and *(c)* there may exist untapped potential in existing training-free NAS, which probably can be unveiled through a unified theoretical understanding. To this end, this paper presents a unified theoretical analysis of gradient-based training-free NAS, which allows us to *(a)* theoretically study their relationships, *(b)* theoretically guarantee their generalization performances, and *(c)* exploit our unified theoretical understanding to develop a novel framework named *hybrid NAS* (HNAS) which consistently boosts training-free NAS in a principled way. Remarkably, HNAS can enjoy the advantages of both training-free (i.e., superior search efficiency) and training-based (i.e., remarkable search effectiveness) NAS, which we have demonstrated through extensive experiments.

1 Introduction

Recent years have witnessed a surging interest in applying *deep neural networks* (DNNs) in real-world applications, e.g., machine translation [1], object detection [2], among others. In order to achieve compelling performances in these applications, many domain-specific neural architectures have been handcrafted by human experts with considerable efforts. However, these efforts have gradually become unaffordable due to the growing demand for customizing neural architectures for different tasks. To this end, *neural architecture search* (NAS) [3] has been proposed to design neural architectures automatically. While many *training-based* NAS algorithms [4, 5] have achieved state-of-the-art (SOTA) performances in various tasks, their search costs usually are unaffordable in resource-constrained scenarios mainly due to their requirement for training DNNs during the search. As a result, a number of training-free metrics have been developed to realize *training-free NAS* [6, 7]. Surprisingly, these training-free NAS algorithms are able to achieve competitive empirical performances even compared with other training-based NAS algorithms while incurring significantly reduced search costs. Moreover, the architectures selected by these training-free NAS algorithms have been empirically found to transfer well to different tasks [7, 8].

Despite the impressive empirical performances of the NAS algorithms using training-free metrics, *a unified theoretical analysis of these training-free metrics* is still lacking in the literature, leading to a few significant implications. Firstly, the theoretical relationships of these training-free metrics are unclear, making it challenging to explain *why they usually lead to comparable empirical results* [9]. Secondly, there is no theoretical guarantee for the empirically observed compelling performances of the architectures selected by NAS algorithms using these training-free metrics. As a consequence,

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

the reason *why NAS using these training-free metrics works well* is still not well understood, and hence there lacks theoretical assurances for NAS practitioners when deploying these algorithms. To the best of our knowledge, the theoretical aspect of NAS with training-free metrics has only been preliminarily studied by Shu et al. [8]. However, their analyses are only based on the training rather than generalization performances of different architectures and are restricted to a *single* training-free metric. Thirdly, there may exist untapped potential in existing training-free NAS algorithms, which probably can be unveiled through a unified theoretical understanding of their training-free metrics.

To this end, we perform a unified theoretical analysis of *gradient-based* training-free NAS to resolve all the three problems discussed above in this paper. Firstly, we theoretically prove the connections among different gradient-based training-free metrics in Sec. 4.1. Secondly, based on these provable connections, we derive a generalization bound for Deep Neural Networks (DNNs) with these training-free metrics and then use them to provide principled interpretations for the compelling empirical performances of existing training-free NAS algorithms (Secs. 4.2 and 4.3). Moreover, we demonstrate that our theoretical interpretation for training-free NAS algorithms, surprisingly, displays the same preference of architecture topology (i.e., wide or deep) as training-based NAS algorithms under certain conditions (Sec. 4.4), which helps to justify the practicality of our theoretical interpretations. Thirdly, by exploiting our unified theoretical analysis, we develop a novel NAS framework named *hybrid NAS* (HNAS) to consistently boost existing training-free NAS algorithms (Sec. 5) in a principled way. Remarkably, through a theory-inspired combination with *Bayesian optimization* (BO), our HNAS framework enjoys the advantages of both training-based (i.e., remarkable search effectiveness) and training-free (i.e., superior search efficiency) NAS simultaneously, making it more advanced than existing training-free and training-based NAS algorithms. Lastly, we use extensive experiments to verify the insights derived from our unified theoretical analysis, as well as the search effectiveness and efficiency of our non-trivial HNAS framework (Sec. 6).

2 Related Works

Recently, a number of training-free metrics have been proposed to estimate the generalization performances of neural architectures, allowing the model training in NAS to be completely avoided. For instance, Mellor et al. [6] have developed a heuristic metric using the overlap of activations in an initialized DNN. Meanwhile, Abdelfattah et al. [9] have empirically revealed a large correlation between training-free metrics that were formerly applied in network pruning, e.g., SNIP [10] and GraSP [11], and the generalization performances of candidate architectures in the search space. These results hence indicate the feasibility of using training-free metrics to estimate the performances of candidate architectures in NAS. Chen et al. [7] have proposed a heuristic metric to trade off the trainability and expressibility of neural architectures in order to find well-performing architectures in various NAS benchmarks. Xu et al. [12] have applied the mean of the Gram matrix of gradients to quickly estimate the performances of architectures. More recently, Shu et al. [8] have employed the theory of *Neural Tangent Kernel* (NTK) [13] to formally derive a performance estimator using the trace norm of the NTK matrix with initialized model parameters, which, surprisingly, is shown to be data- and label-agnostic. Though these existing works have demonstrated the feasibility of training-free NAS through their compelling empirical results, the reason as to *why training-free NAS performs well in practice* and the answer to the question of *how training-free NAS can be further boosted* remain mysteries in the literature. This paper aims to provide theoretically grounded answers to these two questions through a unified analysis of existing gradient-based training-free metrics.

3 Notations and Backgrounds

3.1 Neural Tangent Kernel

To simplify the analysis in this paper, we consider a L-layer DNN with identical widths $n_1 = \cdots = n_{L-1} = n$ and scalar output (i.e., $n_L = 1$) based on the formulation of DNNs in [13]. Let $f(\boldsymbol{x}, \boldsymbol{\theta})$ be the output of a DNN with input $\boldsymbol{x} \in \mathbb{R}^{n_0}$ and parameters $\boldsymbol{\theta} \in \mathbb{R}^d$ that are initialized using the standard normal distribution, the NTK matrix $\boldsymbol{\Theta} \in \mathbb{R}^{m \times m}$ over a dataset of size m is defined as

$$\boldsymbol{\Theta}(\boldsymbol{x}, \boldsymbol{x}'; \boldsymbol{\theta}) = \nabla_{\boldsymbol{\theta}} f(\boldsymbol{x}, \boldsymbol{\theta})^\top \nabla_{\boldsymbol{\theta}} f(\boldsymbol{x}', \boldsymbol{\theta}) \ . \tag{1}$$

Jacot et al. [13] have shown that this NTK matrix $\boldsymbol{\Theta}$ will finally converge to a deterministic form $\boldsymbol{\Theta}_\infty$ in the infinitely wide DNN model. Meanwhile, Arora et al. [14], Allen-Zhu et al. [15] have

further proven that a similar result, i.e., $\mathbf{\Theta} \approx \mathbf{\Theta}_\infty$, can also be achieved in over-parameterized DNNs of finite width. Besides, Arora et al. [14], Lee et al. [16] have revealed that the training dynamics of DNNs can be well-characterized using this NTK matrix at initialization (i.e., $\mathbf{\Theta}_0$ based on the initialized model parameters $\boldsymbol{\theta}_0$) under certain conditions. More recently, Yang and Littwin [17] have further demonstrated that these conclusions about NTK matrix shall also hold for DNNs with any reasonable architecture, even including recurrent neural networks (RNNs) and graph neural networks (GNNs). Therefore, the conclusions drawn based on the formulation above in this paper shall also be applied to the NAS search spaces with complex architectures, which we validate empirically.

3.2 Gradient-Based Training-Free Metrics for NAS

In this paper, we mainly focus on the study of those *gradient-based* training-free metrics, i.e., the training-free metrics that are derived from the gradients of initialized model parameters, which we introduce below. Previous works have empirically shown that better model performances are usually associated with larger values of these training-free metrics in practice [9].

Gradient norm of initialized model parameters. While Abdelfattah et al. [9] were the first to employ the gradient norm of initialized model parameters to estimate the generalization performance of candidate architectures, the same form has also been derived by Shu et al. [8] to approximate their training-free metric efficiently. Following the notations in Sec. 3.1, let $\ell(\cdot,\cdot)$ be the loss function, we define the gradient norm over dataset $S = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^m$ as

$$\mathcal{M}_{\text{Grad}} \triangleq \left\| \frac{1}{m} \sum_{i=1}^{m} \nabla_{\boldsymbol{\theta}} \ell(f(\boldsymbol{x}_i, \boldsymbol{\theta}_0), y_i) \right\|_2 . \tag{2}$$

SNIP and GraSP. SNIP [10] and GraSP [11] were originally proposed for training-free network pruning, and Abdelfattah et al. [9] have applied them in training-free NAS to estimate the performances of candidate architectures without model training. Following the notations in Sec. 3.1, let $\mathbf{H}_i \in \mathbb{R}^{d\times d}$ denote the hessian matrix induced by input $\boldsymbol{x}_i$, the metrics of SNIP and GraSP on dataset $S = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^m$ can be defined as

$$\mathcal{M}_{\text{SNIP}} \triangleq \left| \frac{1}{m} \sum_{i}^{m} \boldsymbol{\theta}_0^\top \nabla_{\boldsymbol{\theta}} \ell(f(\boldsymbol{x}_i, \boldsymbol{\theta}_0), y_i) \right| , \ \mathcal{M}_{\text{GraSP}} \triangleq \left| \frac{1}{m} \sum_{i}^{m} \boldsymbol{\theta}_0^\top \left(\mathbf{H}_i \nabla_{\boldsymbol{\theta}} \ell(f(\boldsymbol{x}_i, \boldsymbol{\theta}_0), y_i)\right) \right| . \tag{3}$$

Trace norm of NTK matrix at initialization. Recently, Shu et al. [8] have reformulated NAS into a constrained optimization problem to maximize the trace norm of the NTK matrix at initialization. Shu et al. [8] have empirically shown that this trace norm is highly correlated with the generalization performance of candidate architectures under their derived constraint. Let $\mathbf{\Theta}_0$ be the NTK matrix based on initialized model parameters $\boldsymbol{\theta}_0$ of a DNN, then without considering the constraint in [8], we frame this training-free metric on dataset $S = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^m$ as

$$\mathcal{M}_{\text{Trace}} \triangleq \sqrt{\|\mathbf{\Theta}_0\|_{\text{tr}}/m} . \tag{4}$$

4 Theoretical Analyses of Training-Free NAS

4.1 Connections among Training-Free Metrics

Notably, though the gradient-based training-free metrics introduced in Sec. 3.2 seem to have distinct mathematical forms, most of them will actually achieve similar empirical performances in practice [9]. More interestingly, these metrics in fact share the similarity of using the gradients of initialized model parameters in their calculations. Based on these facts, we propose the following hypothesis to explain the similar performances achieved by different training-free metrics in Sec. 3.2: *The training-free metrics in Sec. 3.2 may be theoretically connected and hence could provide similar characterization for the generalization performances of architectures*. We validate this hypothesis affirmatively and use the following theorem to establish the theoretical connections among these metrics.

Theorem 1. *Let the loss function $\ell(\cdot,\cdot)$ in gradient-based training-free metrics be β-Lipschitz continuous and γ-Lipschitz smooth in the first argument. There exist the constants $C_1, C_2, C_3 > 0$, with a high probability,*

$$\mathcal{M}_{\text{Grad}} \leq C_1 \mathcal{M}_{\text{Trace}}, \ \mathcal{M}_{\text{SNIP}} \leq C_2 \mathcal{M}_{\text{Trace}}, \ \mathcal{M}_{\text{GraSP}} \leq C_3 \mathcal{M}_{\text{Trace}} .$$

The proof of Theorem 1 are given in Appendix A.1. Notably, our Theorem 1 implies that with a high probability, architectures of larger $\mathcal{M}_{\text{Grad}}$, $\mathcal{M}_{\text{SNIP}}$ or $\mathcal{M}_{\text{Grad}}$ will also achieve a larger $\mathcal{M}_{\text{Trace}}$ given the inequalities above. That is, the value of $\mathcal{M}_{\text{Grad}}$, $\mathcal{M}_{\text{SNIP}}$ and $\mathcal{M}_{\text{Grad}}$ for different architectures in the NAS search space should be highly correlated with the value of $\mathcal{M}_{\text{Trace}}$. As a consequence, these training-free metrics should be able to provide similar estimation of the generalization performances of architectures (validated in Sec. 6.2) and hence similar performances can be achieved when using these metrics (validated in Sec. 6.4). Overall, the training-free NAS metrics from Sec. 3.2 can all be theoretically connected with $\mathcal{M}_{\text{Trace}}$ despite their distinct mathematical forms. Note that though our Theorem 1 is only able to establish the theoretical connections between $\mathcal{M}_{\text{Trace}}$ and other training-free metrics, our empirical results in Appendix C.1 further reveal that any two training-free metrics from Sec. 3.2 will also be highly correlated. Interestingly, these results serve as principled justifications for the similar performances achieved by these training-free metrics in [9].

4.2 A Generalization Bound Induced by Training-free Metrics

Let dataset $S{=}\{(\boldsymbol{x}_i, y_i)\}_{i=1}^m$ be randomly sampled from a distribution $\mathcal{D}$, we denote $\mathcal{L}_S(\cdot)$ as the training error on S and $\mathcal{L}_{\mathcal{D}}(\cdot)$ as the corresponding *generalization error* on $\mathcal{D}$. Intuitively, a smaller generalization error indicates a better *generalization performance*. Thanks to the common theoretical underpinnings of gradient-based training-free metrics formalized by Theorem 1, we can perform a *unified generalization analysis* for DNNs in terms of these metrics by making use of the NTK theory [13]. Define $\ell(f, y) \triangleq (f-y)^2/2$ and $\eta_0 \triangleq \min\{2n^{-1}(\lambda_{\min}(\boldsymbol{\Theta}_\infty)+\lambda_{\max}(\boldsymbol{\Theta}_\infty))^{-1}, m\lambda_{\max}^{-1}(\boldsymbol{\Theta}_0)\}$ with $\lambda_{\min}(\cdot), \lambda_{\max}(\cdot)$ being the minimum and maximum eigenvalue of a matrix respectively, we derive the following theorem:

Theorem 2. *Assume $\|\boldsymbol{x}_i\|_2 \leq 1$ and $f(\boldsymbol{x}_i, \boldsymbol{\theta}_0), \lambda_{\min}(\boldsymbol{\Theta}_0), y_i \in [0, 1]$ for any $(\boldsymbol{x}_i, y_i) \in S$. There exists a constant $N \in \mathbb{N}$ such that for any $n > N$, when applying gradient descent with learning rate $\eta < \eta_0$, the generalization error of f_t at time $t > 0$ can be bounded as below with a high probability,*

$$\mathcal{L}_{\mathcal{D}}(f_t) \leq \mathcal{L}_S(f_t) + \mathcal{O}(\kappa/\mathcal{M}) \ .$$

Here, $\mathcal{M}$ can be any metric in Sec. 3.2 and $\kappa \triangleq \lambda_{\max}(\boldsymbol{\Theta}_0)/\lambda_{\min}(\boldsymbol{\Theta}_0)$ is the condition number of $\boldsymbol{\Theta}_0$.

Its proof is in Appendix A.2 and the second term $\mathcal{O}(\kappa/\mathcal{M})$ in Theorem 2 represents the *generalization gap* of DNN models. Notably, our Theorem 2 provides an explicit theoretical connection between the gradient-based training-free metrics from Sec. 3.2 and the generalization gap of DNNs, which later serves as the foundation to theoretically interpret the compelling performances achieved by existing training-free NAS algorithms (Sec. 4.3). Compared to the traditional Rademacher complexity [18], these training-free metrics provide alternative methods to measure the complexity of DNNs when estimating the generalization gap of DNNs.

4.3 Concrete Generalization Guarantees for Training-Free NAS

Since the first term $\mathcal{L}_S(\cdot)$ in our Theorem 2 may also depend on the training-free metric $\mathcal{M}$, it also needs to be taken into account when analyzing the generalization performance (or generalization error $\mathcal{L}_{\mathcal{D}}(\cdot)$) for training-free NAS methods. To this end, in this section, we derive concrete generalization guarantees for NAS methods using training-free metrics by considering two different scenarios (i.e., the *realizable* and *non-realizable* scenarios) for the training error term $\mathcal{L}_S(\cdot)$ in Theorem 2, which finally give rise to principled interpretations for different training-free NAS methods [9, 8, 7].

The realizable scenario. Similar to [18], we assume that a zero training error (i.e., $\mathcal{L}_S(f_t) \to 0$ when t is sufficiently large) can be achieved in the realizable scenario. By further assuming that the condition number κ in Theorem 2 is bounded by κ_0 for all candidate architectures in the search space, we can then derive the following generalization guarantee (Corollary 1) for the realizable scenario.

Corollary 1. *Under the conditions in Theorem 2, for f_t at convergence (i.e., $t \to \infty$) in the realizable scenario and for any training-free metric $\mathcal{M}$ from Sec. 3.2, the following holds with a high probability,*

$$\mathcal{L}_{\mathcal{D}}(f_t) \leq \mathcal{O}(1/\mathcal{M}) \ .$$

Corollary 1 is obtained by introducing $\mathcal{L}_S(f_t) = 0$ and $\kappa \leq \kappa_0$ into Theorem 2. Importantly, Corollary 1 suggests that in the realizable scenario, the generalization error of DNNs is negatively correlated

with the metrics from Sec. 3.2. That is, an architecture with a larger value of training-free metric $\mathcal{M}$ generally achieves a better generalization performance. This implies that in order to select well-performing architectures, we can simply maximize $\mathcal{M}$ to find $\mathcal{A}^* = \arg\max_{\mathcal{A}} \mathcal{M}(\mathcal{A})$ where $\mathcal{A}$ denotes any architecture in the search space. Interestingly, this formulation aligns with the training-free NAS method from [9], which has made use of the metrics $\mathcal{M}_{\text{Grad}}$, $\mathcal{M}_{\text{SNIP}}$ and $\mathcal{M}_{\text{GraSP}}$ to achieve good empirical performances. Therefore, our Corollary 1 provides a valid generalization guarantee and also a principled justification for the method from [9].

The non-realizable scenario. In practice, different candidate architectures in a NAS search space typically have diverse non-zero training errors [8] and κ [7]. Therefore, the assumptions of the zero training error and the bounded κ in the realizable scenario above may be impractical. In light of this, we drop these two assumptions and derive the following generalization guarantee (Corollary 2) for the non-realizable scenario, which, interestingly, facilitates theoretically grounded interpretations for the training-free NAS methods from [8, 7].

Corollary 2. *Under the conditions in Theorem 2, for any f_t at time $t > 0$ and any training-free metric $\mathcal{M}$ from Sec. 3.2 in the non-realizable scenario, there exists a constant $c > 0$ such that with a high probability,*

$$\mathcal{L}_{\mathcal{D}}(f_t) \leq \frac{1}{2} m \left(1 - \eta \mathcal{M}^2/(mc)\right)^{2t} + \mathcal{O}(\kappa/\mathcal{M}) \; .$$

Its proof is given in Appendix A.3. Notably, our Corollary 2 suggests that when $\mathcal{M} \in [0, \sqrt{mc/\eta}]$, an architecture with a larger value of the metric $\mathcal{M}$ will lead to a better generalization performance because such a model has both a faster convergence (i.e., the first term decreases faster w.r.t time t) and a smaller generalization gap (i.e., the second term is smaller). Interestingly, Shu et al. [8] have leveraged this insight to introduce the training-free metric of $\mathcal{M}_{\text{Trace}}$ with a constraint, which has achieved a higher correlation with the generalization performance of architectures than the metrics from [9]. This therefore implies that our Corollary 2 followed by [8] provides a better characterization of the generalization performance of architectures than Corollary 1 followed by [9] since the non-realizable scenario we have considered will be more realistic than the realizable scenario as explained above. Meanwhile, Corollary 2 also suggests that there exists a trade-off in terms of $\mathcal{M}$ between the model convergence (i.e., the first term) and the generalization gap (i.e., the second term) when $\mathcal{M} > \sqrt{mc/\eta}$, which surprisingly is similar to the empirically motivated trainability and expressivity trade-off in [7]. In addition, Corollary 2 also indicates that for architectures achieving similar values of $\mathcal{M}$, the ones with smaller condition numbers κ generally achieve better generalization performance. Interestingly, such a result also aligns with the conclusion from [7]. Therefore, our Corollary 2 also provides a principled justification for the training-free NAS method in [7].

4.4 Connection to Architecture Topology

Interestingly, we can even prove that the condition number κ in our Corollary 2 is theoretically related to the architecture topology, i.e., whether the architecture is wide (and shallow) or deep (and narrow). Inspired by the analysis from [19], we firstly analyze the eigenvalues of the NTK matrices of two different architecture topologies (i.e., wide vs. deep architectures), which gives us an insight into the difference between their corresponding κ. We mainly consider the following wide (i.e., f) and deep (i.e., f') architecture illustrated in Figure 3, respectively:

$$f(\boldsymbol{x}) = \mathbf{1}^\top \textstyle\sum_{i=1}^{L} \mathbf{W}^{(i)} \boldsymbol{x} \; , \; f'(\boldsymbol{x}) = \mathbf{1}^\top (\prod_{i=1}^{L} \mathbf{W}^{(i)}) \boldsymbol{x} \tag{5}$$

where $\mathbf{W}^{(i)} \in \mathbb{R}^{n \times n}$ for any $i \in \{1, \cdots, L\}$ and every element of $\mathbf{W}^{(i)}$ is independently initialized using the standard normal distribution. Here, $\mathbf{1}$ denotes an n-dimensional vector with every element being one. Let $\boldsymbol{\Theta}_0$ and $\boldsymbol{\Theta}'_0$ be the NTK matrices of f and f' that are evaluated on the finite dataset $S = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^m$, respectively, we derive the following theorem:

Theorem 3. *Let dataset S be normalized using its statistic mean and covariance such that $\mathbb{E}[\boldsymbol{x}] = 0$ and $\mathbf{X}^\top \mathbf{X} = \mathbf{I}$ given $\mathbf{X} \triangleq [\boldsymbol{x}_1 \boldsymbol{x}_2 \cdots \boldsymbol{x}_m]$, we have*

$$\boldsymbol{\Theta}_0 = Ln \cdot \mathbf{I} \; , \; \mathbb{E}\left[\boldsymbol{\Theta}'_0\right] = Ln^L \cdot \mathbf{I} \; .$$

Its proof is in Appendix A.4. Notably, Theorem 3 shows that the NTK matrix of the wide architecture in (5) is guaranteed to be a scaled identity matrix, whereas the NTK matrix of the deep architecture

Algorithm 1: Hybrid Neural Architecture Search (HNAS)

Input: Training and validation dataset, metric $\mathcal{M}$ evaluated on architecture pool $\mathcal{P}$, and $F(\cdot)$
for step $k = 1, \ldots, K$ **do**
  Choose μ_k, ν_k using BO algorithm based on its exploration and exploitation trade-off
  Obtain the optimal candidate $\mathcal{A}_k^*$ in $\mathcal{P}$ by solving (7)
  Evaluate the validation performance of $\mathcal{A}_k^*$
  Update the GP surrogate applied in the BO algorithm
end for
Select the final architecture $\mathcal{A}^*$ achieving the best validation performance among $\{\mathcal{A}_k^*\}_{k=1}^K$.

in (5) is a scaled identity matrix *only in expectation* over random initialization. Consequently, we always have $\kappa = 1$ for the initialized wide architecture, while $\kappa > 1$ with high probability for the initialized deep architecture. Also note that as we have discussed in Sec. 4.3, our Corollary 2 shows that (given similar values of $\mathcal{M}$) an architecture with a smaller κ is likely to generalize better. Therefore, this implies that wide architectures generally achieve better generalization performance than deep architectures (given similar values of $\mathcal{M}$). This, surprisingly, aligns with the findings from [19] which shows that wide architectures are preferred in *training-based NAS* due to their competitive performances in practice. More interestingly, based on the definition of $\mathcal{M}_{\text{Trace}}$ (4), Theorem 3 also indicates that deep architectures are expected to have larger values of $\mathcal{M}_{\text{Trace}}$ (due to the larger scale of $\mathbb{E}\left[\mathbf{\Theta}_0'\right]$ for deep architectures) and hence achieve larger model complexities than wide architectures.

5 Hybrid Neural Architecture Search

5.1 A Unified Objective for Training-Free NAS

Our theoretical understanding of training-free NAS in Sec. 4 finally allows us to address the following question in a principled way: *How can we consistently boost existing training-free NAS algorithms?* Specifically, to realize this target, we propose to select well-performing architectures by minimizing the upper bound on the generalization error in Corollary 2 given any training-free metric from Sec. 3.2. We expect this choice to lead to improved performances over the method from [9] because Corollary 2 provides a more practical generalization guarantee for training-free NAS than Corollary 1 followed by [9] (Sec. 4.3). Formally, let $\mathcal{A}$ be any architecture in the search space and let $\mathcal{M}$ be any training-free metric from Sec. 3.2, then the NAS problem can be formulated below in a unified manner:

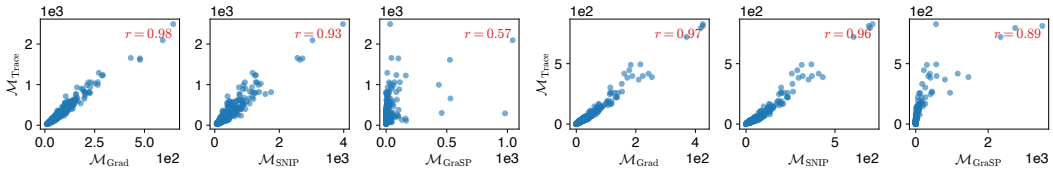
$$\min_{\mathcal{A}} \frac{1}{2} m \left(1 - \frac{\eta}{mc} \mathcal{M}^2(\mathcal{A})\right)^{2t} + \mathcal{O}\left(\frac{\kappa(\mathcal{A})}{\mathcal{M}(\mathcal{A})}\right) . \tag{6}$$

We further reformulate (6) into the following form:

$$\min_{\mathcal{A}} \kappa(\mathcal{A})/\mathcal{M}(\mathcal{A}) + \mu F(\mathcal{M}^2(\mathcal{A}) - \nu) \tag{7}$$

where $F(x) \triangleq x^{2t}$, and μ and ν are hyperparameters we introduced to absorb the impact of all other parameters in (6). Compared with the diverse form of NAS objectives in [9, 7, 8], our (7) proposes a unified form of NAS objectives for all the training-free metrics in Sec. 3.2, making it easier for practitioners to deploy NAS with different training-free metrics. Note that our NAS objective in (7) is a natural consequence of our generalization guarantee in Corollary 2 and therefore will be more theoretically grounded, in contrast to the empirically motivated objective in [7]. Moreover, our (7) advances the training-free NAS method based on $\mathcal{M}_{\text{Trace}}$ from [8], because our (7) *(a)* is derived from the generalization error instead of the training error (that is followed by [8]) of DNNs, which therefore will be more sound and practical, *(b)* have unified all the training-free metrics from Sec. 3.2, and *(c)* have considered the impact of condition number κ which is shown to be critical in practice (see our Appendix C.2). Above all, our unified NAS objective in (7) is expected to be able to lead to improved performances over other existing training-free NAS methods.

5.2 Optimization and Search Algorithm



(a) NAS-Bench-101 (b) NAS-Bench-201

Figure 1: Spearman correlation between $\mathcal{M}_{\text{Trace}}$ and other training-free metrics from Sec. 3.2, which are evaluated in NAS-Bench-101/201. The correlation coefficient r is given in the corner of each plot.

Our theoretically motivated NAS objective in (7) has unified all training-free metrics from Sec. 3.2 and improved over existing training-free NAS methods. However, its practical deployment requires the determination of the hyperparameters μ and ν,[1] which can be non-trivial in practice. To this end, we further introduce *Bayesian optimization* (BO) [20] to optimize the hyperparameters μ and ν in order *to maximize the true validation performance* of the architectures selected by different μ and ν. Specifically, BO uses a Gaussian process (GP) as a surrogate to model the objective function (i.e., the validation performance) in order to sequentially choose the queried inputs (i.e., the values of μ and ν). This finally completes our theoretically grounded NAS framework called *hybrid NAS* (HNAS), which novelly unifies all training-free metrics from Sec. 3.2 (Algorithm 1). In every iteration k of HNAS, we firstly select the optimal candidate $\mathcal{A}_k^*$ by maximizing our training-free NAS objective in (7) using the values of μ and ν queried by BO in the current iteration. Next, we evaluate the validation performance of $\mathcal{A}_k^*$ and then use it to update the GP surrogate, which will be used to choose the values of μ and ν in the next iteration. After HNAS completes, the final selected architecture is chosen as the one achieving the best validation performance among all the optimal candidates. Thanks to the use of validation performance as the objective for BO, our HNAS is expected to be able to enjoy the advantages of both training-free (i.e., superior search efficiency) and training-based NAS (i.e., remarkable search effectiveness) as supported by our extensive empirical results in Sec. 6.4. More interestingly, by novelly introducing BO to optimize the low-dimensional continuous hyperparameters μ and ν rather than the high-dimensional discrete architectural hyperparameters in the NAS search space, our HNAS is able to avoid the high-dimensional discrete optimization issues that standard BO algorithms usually attain when directly optimizing in the NAS search space, allowing our HNAS to be more efficient and effective in practice as supported in our Sec. 6.4.

6 Experiments

6.1 Connections among Training-Free Metrics

We firstly validate the theoretical connections between $\mathcal{M}_{\text{Trace}}$ and other training-free metrics from Sec. 3.2 by examining their Spearman correlations for all architectures in NAS-Bench-101 [21] and NAS-Bench-201 [22] using CIFAR-10 [23]. Figure 1 illustrates the result where all these training-free metrics are evaluated using a batch (with size 64) of sampled data following that of [9]. [2] Notably, the results in Figure 1 demonstrate that $\mathcal{M}_{\text{Trace}}$ and other training-free metrics from Sec. 3.2 are indeed highly correlated since they consistently achieve high positive correlations in different search spaces. These empirical results indeed align with the interpretation of our Theorem 1 in Sec. 4.1. The correlation between any two training-free metrics from Sec. 3.2 is in Appendix C.1, which shows that all training-free metrics from Sec. 3.2 are also highly correlated. Above all, besides Theorem 1, our empirical results have also justified the connections among all the training-free metrics from Sec. 3.2.

6.2 Generalization Guarantees for Training-Free NAS

We then verify the validity of our generalization guarantees for training-free NAS (Sec. 4.3) by examining the correlation between the generalization bound in the realizable (Corollary 1) or non-realizable (Corollary 2) scenario and the test errors of architectures in NAS-Bench-101/201. Similar to HNAS (Algorithm 1), we use BO with hundreds of iterations to decide the non-trivial parameters in Corollary 2. Table 1 summarizes the results on CIFAR-10 where a higher positive correlation

[1] We usually set $t = 1$ in practice.

[2] Note that we follow the same approach to evaluate these training-free metrics in the following sections.

Table 1: Correlation coefficient between the test errors evaluated on CIFAR-10 and the generalization guarantees in Sec. 4.3 for the architectures in NAS-Bench-101/201.

Metric	NAS-Bench-101		NAS-Bench-201	
	Spearman	Kendall's Tau	Spearman	Kendall's Tau
Realizable scenario				
$\mathcal{M}_{\text{Grad}}$	−0.25	−0.17	0.64	0.47
$\mathcal{M}_{\text{SNIP}}$	−0.21	−0.15	0.64	0.47
$\mathcal{M}_{\text{GraSP}}$	−0.45	−0.31	0.57	0.40
$\mathcal{M}_{\text{Trace}}$	−0.30	−0.21	0.54	0.39
Non-realizable scenario				
$\mathcal{M}_{\text{Grad}}$	0.35	0.23	0.75	0.56
$\mathcal{M}_{\text{SNIP}}$	0.37	0.25	0.75	0.56
$\mathcal{M}_{\text{GraSP}}$	0.46	0.32	0.69	0.50
$\mathcal{M}_{\text{Trace}}$	0.33	0.23	0.70	0.51

Table 2: Comparison of topology, $\mathcal{M}_{\text{Trace}}$ & κ of different architectures. The topology width/depth of each architecture is followed by the maximum value in the search space (separated by a slash).

Architecture	Topology		$\mathcal{M}_{\text{Trace}}$	κ
	Width	Depth		
NASNet	5.0/5.0	2/6	31±2	118±41
AmoebaNet	4.0/5.0	4/6	36±2	110±39
ENAS	5.0/5.0	2/6	36±2	98±33
DARTS	3.5/4.0	3/5	33±2	122±58
SNAS	4.0/4.0	2/5	31±2	126±47
WIDE	4.0/4.0	2/5	27±1	141±36
DEEP	**1.5**/4.0	**5**/5	**131±16**	**209±107**

implies a better agreement between our generalization guarantee and the generalization performance of architectures. As shown, the generalization bound in the realizable scenario performs a compelling characterization of the test errors in NAS-Bench-201 by achieving large positive correlations, whereas it fails to provide a precise characterization in a larger search space, i.e., NAS-Bench-101. Fortunately, our generalization bound in the non-realizable scenario performs consistent improvement over it by obtaining higher positive correlations. These results thus imply that Corollary 1 may only provide a good characterization for training-free NAS in small-scale search spaces, whereas our Corollary 2 can be more valid and robust in practice. As a result, our (6) following Corollary 2 should indeed be able to improve over the NAS objective in [9] following Corollary 1 as justified in our Sec. 5 above. Meanwhile, the comparable results achieved by all training-free metrics from Sec. 3.2 further validate the connections among these metrics (Theorem 1). Moreover, the results in Appendix C.2 further confirm the validity and practicality of our generalization guarantees for training-free NAS.

6.3 Connection to Architecture Topology

To validate the theoretical connections between architecture topology (wide vs. deep) and the value of training-free metric $\mathcal{M}_{\text{Trace}}$ as well as the condition number κ shown in Sec. 4.4, we compare the topology width/depth, $\mathcal{M}_{\text{Trace}}$ and κ of the architectures selected by different state-of-the-art training-based NAS algorithms in the DARTS search space, including NASNet [24], AmoebaNet [25], ENAS [4], DARTS [5], and SNAS [26]. Table 2 summarizes the results where we apply the same definition of topology width/depth in [19] (more details in [19]). We also include the widest (called WIDE) and the deepest (called DEEP) architecture in the DARTS search space into this comparison. As shown, wide architectures (i.e., all architectures except DEEP in Table 2) consistently achieve lower condition number κ and smaller values of $\mathcal{M}_{\text{Trace}}$ than deep architecture (i.e., DEEP), which aligns with our theoretical insights in Sec. 4.4.

6.4 Effectiveness and Efficiency of HNAS

To justify that our theoretically motivated HNAS is able to enjoy the advantages of both training-free (i,e., the superior search efficiency) and training-based (i.e., the remarkable search effectiveness) NAS, we compare it with other baselines in NAS-Bench-201 (Table 3). HNAS, surprisingly, advances both training-based and training-free baselines by consistently selecting architectures achieving the best performances, leading to smaller gaps toward the optimal test errors in the search space. Meanwhile, HNAS requires at most $13\times$ lower search costs than training-based NAS algorithms, which is even smaller than the training-free baseline KNAS. Moreover, thanks to the superior evaluation efficiency of training-free metrics, HNAS can be deployed efficiently in not only continuous (where search space is represented as a supernet) but also discrete search space. As for NAS under limited search budgets (Figure 2), HNAS also advances all other baselines by achieving improved search efficiency and effectiveness. Appendix C.4 further includes the impressive search results achieved by HNAS on CIFAR-10/100 and ImageNet in the DARTS search space. Overall, HNAS is indeed able to enjoy the advantages of both training-free (i.e., the superior search efficiency) and training-based NAS (i.e., the remarkable search effectiveness), which consistently boosts existing training-free NAS methods.

Table 3: Comparison among NAS algorithms in NAS-Bench-201. The result of HNAS is reported with the mean and standard deviation of 5 independent searches and its search costs are evaluated on a Nvidia 1080Ti. C & D in the last column denote continuous and discrete search space, respectively.

Architecture	**Test Accuracy (%)**			**Cost** (GPU Sec.)	**Method**	**Applicable Space**
	C10	C100	IN-16			
ResNet [27]	93.97	70.86	43.63	-	manual	-
REA†	93.92±0.30	71.84±0.99	45.15±0.89	12000	evolution	C & D
RS (w/o sharing)†	93.70±0.36	71.04±1.07	44.57±1.25	12000	random	C & D
REINFORCE†	93.85±0.37	71.71±1.09	45.24±1.18	12000	RL	C & D
BOHB†	93.61±0.52	70.85±1.28	44.42±1.49	12000	BO+bandit	C & D
ENAS‡ [4]	93.76±0.00	71.11±0.00	41.44±0.00	15120	RL	C
DARTS (1st)‡ [5]	54.30±0.00	15.61±0.00	16.32±0.00	16281	gradient	C
DARTS (2nd)‡ [5]	54.30±0.00	15.61±0.00	16.32±0.00	43277	gradient	C
GDAS‡ [28]	93.44±0.06	70.61±0.21	42.23±0.25	8640	gradient	C
NASWOT [6]	92.96±0.81	69.98±1.22	44.44±2.10	306	training-free	C & D
TE-NAS [7]	93.90±0.47	71.24±0.56	42.38±0.46	1558	training-free	C
KNAS [12]	93.05	68.91	34.11	4200	training-free	C & D
NASI [8]	93.55±0.10	71.20±0.14	44.84±1.41	120	training-free	C
HNAS ($\mathcal{M}_{\text{Grad}}$)	**94.04**±0.21	71.75±1.04	**45.91**±0.88	3010	hybrid	C & D
HNAS ($\mathcal{M}_{\text{SNIP}}$)	**93.94**±0.02	71.49±0.11	**46.07**±0.14	2976	hybrid	C & D
HNAS ($\mathcal{M}_{\text{GraSP}}$)	**94.13**±0.13	**72.59**±0.82	**46.24**±0.38	3148	hybrid	C & D
HNAS ($\mathcal{M}_{\text{Trace}}$)	**94.07**±0.10	**72.30**±0.70	**45.93**±0.37	3006	hybrid	C & D
Optimal	94.37	73.51	47.31	-	-	-

† Reported by Dong and Yang [22].
‡ Re-evaluated for 5 times using the codes provided by Dong and Yang [22].

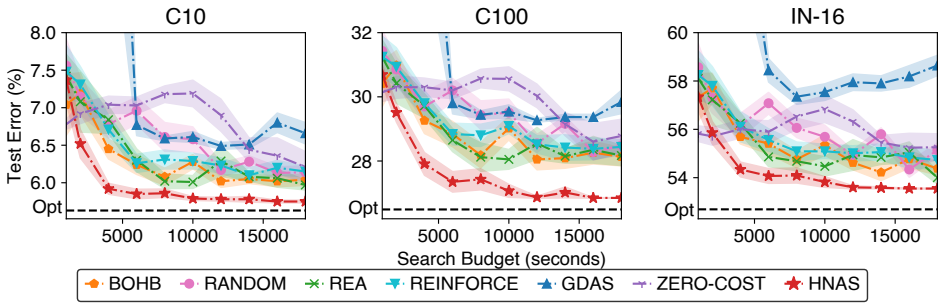


Figure 2: Comparison between HNAS ($\mathcal{M}_{\text{Trace}}$) and other NAS baselines in NAS-Bench-201 under varying search budgets. Here, the ZERO-COST method is borrowed from [9] by using $\mathcal{M}_{\text{Trace}}$ and each algorithm is reported with the mean and standard error of ten independent searches.

7 Conclusion & Discussion

This paper performs a unified theoretical analysis of NAS algorithms using gradient-based training-free metrics, which allows us to *(a)* theoretically unveil the connections among these training-free metrics, *(b)* provide theoretical guarantees for the empirically observed compelling performance of these training-free NAS algorithms, and *(c)* exploit these theoretical understandings to develop a novel framework called HNAS that can consistently boost existing training-free NAS. We expect that our theoretical understanding could provide valuable prior knowledge for the design of training-free metrics and NAS search space in the future. Moreover, considering the close relationship between network pruning and NAS, we expect our unified theoretical analyses to be capable of inspiring more theoretical understanding and improvement over existing training-free network pruning algorithms in the literature. More importantly, the impressive performance achieved by our HNAS framework is expected to be able to encourage more attention to the integration of training-free and training-based approaches in other machine learning fields in order to enjoy the advantages of these two types of methods simultaneously.

References

[1] Felix Stahlberg. Neural machine translation: A review and survey. 2020.

[2] Zhong-Qiu Zhao, Peng Zheng, Shou-tao Xu, and Xindong Wu. Object detection with deep learning: A review. *IEEE Trans. Neural Networks Learn. Syst.*, 30:3212–3232, 2019.

[3] Barret Zoph and Quoc V. Le. Neural architecture search with reinforcement learning. In *Proc. ICLR*, 2017.

[4] Hieu Pham, Melody Y. Guan, Barret Zoph, Quoc V. Le, and Jeff Dean. Efficient neural architecture search via parameter sharing. In *Proc. ICML*, pages 4092–4101, 2018.

[5] Hanxiao Liu, Karen Simonyan, and Yiming Yang. DARTS: Differentiable architecture search. In *Proc. ICLR*, 2019.

[6] Joseph Mellor, Jack Turner, Amos J. Storkey, and Elliot J. Crowley. Neural architecture search without training. In *Proc. ICML*, 2021.

[7] Wuyang Chen, Xinyu Gong, and Zhangyang Wang. Neural architecture search on imagenet in four gpu hours: A theoretically inspired perspective. In *Proc. ICLR*, 2021.

[8] Yao Shu, Shaofeng Cai, Zhongxiang Dai, Beng Chin Ooi, and Bryan Kian Hsiang Low. NASI: Label- and data-agnostic neural architecture search at initialization. In *Proc. ICLR*, 2022.

[9] Mohamed S. Abdelfattah, Abhinav Mehrotra, Lukasz Dudziak, and Nicholas D. Lane. Zero-cost proxies for lightweight NAS. In *Proc. ICLR*, 2021.

[10] Namhoon Lee, Thalaiyasingam Ajanthan, and Philip H. S. Torr. SNIP: Single-shot network pruning based on connection sensitivity. In *Proc. ICLR*, 2019.

[11] Chaoqi Wang, Guodong Zhang, and Roger B. Grosse. Picking winning tickets before training by preserving gradient flow. In *Proc. ICLR*, 2020.

[12] Jingjing Xu, Liang Zhao, Junyang Lin, Rundong Gao, Xu Sun, and Hongxia Yang. KNAS: Green neural architecture search. In *Proc. ICML*, pages 11613–11625, 2021.

[13] Arthur Jacot, Clément Hongler, and Franck Gabriel. Neural Tangent Kernel: Convergence and generalization in neural networks. In *Proc. NeurIPS*, pages 8580–8589, 2018.

[14] Sanjeev Arora, Simon S. Du, Wei Hu, Zhiyuan Li, Ruslan Salakhutdinov, and Ruosong Wang. On exact computation with an infinitely wide neural net. In *Proc. NeurIPS*, pages 8139–8148, 2019.

[15] Zeyuan Allen-Zhu, Yuanzhi Li, and Zhao Song. A convergence theory for deep learning via over-parameterization. In *Proc. ICML*, pages 242–252, 2019.

[16] Jaehoon Lee, Lechao Xiao, Samuel S. Schoenholz, Yasaman Bahri, Roman Novak, Jascha Sohl-Dickstein, and Jeffrey Pennington. Wide neural networks of any depth evolve as linear models under gradient descent. In *Proc. NeurIPS*, pages 8570–8581, 2019.

[17] Greg Yang and Etai Littwin. Tensor programs IIb: Architectural universality of neural tangent kernel training dynamics. In *Proc. ICML*, pages 11762–11772, 2021.

[18] Mehryar Mohri, Afshin Rostamizadeh, and Ameet Talwalkar. *Foundations of Machine Learning*. Adaptive computation and machine learning. MIT Press, 2012.

[19] Yao Shu, Wei Wang, and Shaofeng Cai. Understanding architectures learnt by cell-based neural architecture search. In *Proc. ICLR*, 2020.

[20] Jasper Snoek, Hugo Larochelle, and Ryan P. Adams. Practical Bayesian optimization of machine learning algorithms. In *Proc. NeurIPS*, pages 2960–2968, 2012.

[21] Chris Ying, Aaron Klein, Eric Christiansen, Esteban Real, Kevin Murphy, and Frank Hutter. NAS-Bench-101: Towards reproducible neural architecture search. In *Proc. ICML*, pages 7105–7114, 2019.

[22] Xuanyi Dong and Yi Yang. NAS-Bench-201: Extending the scope of reproducible neural architecture search. In *Proc. ICLR*, 2020.

[23] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009.

[24] Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V. Le. Learning transferable architectures for scalable image recognition. In *Proc. CVPR*, pages 8697–8710, 2018.

[25] Esteban Real, Alok Aggarwal, Yanping Huang, and Quoc V. Le. Regularized evolution for image classifier architecture search. In *Proc. AAAI*, pages 4780–4789, 2019.

[26] Sirui Xie, Hehui Zheng, Chunxiao Liu, and Liang Lin. SNAS: Stochastic neural architecture search. In *Proc. ICLR*, 2019.

[27] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proc. CVPR*, pages 770–778, 2016.

[28] Xuanyi Dong and Yi Yang. Searching for a robust neural architecture in four GPU hours. In *Proc. CVPR*, pages 1761–1770, 2019.

[29] Pranjal Awasthi, Natalie Frank, and Mehryar Mohri. On the rademacher complexity of linear hypothesis sets. arXiv:2007.11045, 2020.

[30] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A large-scale hierarchical image database. In *Proc. CVPR*, 2009.

[31] Jack Turner, Elliot J. Crowley, Michael F. P. O'Boyle, Amos J. Storkey, and Gavin Gray. BlockSwap: Fisher-guided block substitution for network compression on a budget. In *Proc. ICLR*, 2020.

[32] Hidenori Tanaka, Daniel Kunin, Daniel L. Yamins, and Surya Ganguli. Pruning neural networks without any data by iteratively conserving synaptic flow. In *Proc. NeurIPS*, 2020.

[33] Patryk Chrabaszcz, Ilya Loshchilov, and Frank Hutter. A downsampled variant of imagenet as an alternative to the CIFAR datasets. arXiv:1707.08819, 2017.

[34] Terrance Devries and Graham W. Taylor. Improved regularization of convolutional neural networks with cutout. arXiv:1708.04552, 2017.

[35] Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Q. Weinberger. Densely connected convolutional networks. In *Proc. CVPR*, pages 2261–2269, 2017.

[36] Chenxi Liu, Barret Zoph, Maxim Neumann, Jonathon Shlens, Wei Hua, Li-Jia Li, Li Fei-Fei, Alan L. Yuille, Jonathan Huang, and Kevin Murphy. Progressive neural architecture search. In *Proc. ECCV*, pages 19–35, 2018.

[37] Renqian Luo, Fei Tian, Tao Qin, Enhong Chen, and Tie-Yan Liu. Neural architecture optimization. In *Proc. NeurIPS*, pages 7827–7838, 2018.

[38] Quanming Yao, Ju Xu, Wei-Wei Tu, and Zhanxing Zhu. Efficient neural architecture search via proximal iterations. In *Proc. AAAI*, pages 6664–6671, 2020.

[39] Xin Chen, Lingxi Xie, Jun Wu, and Qi Tian. Progressive differentiable architecture search: Bridging the depth gap between search and evaluation. In *Proc. ICCV*, pages 1294–1303, 2019.

[40] Xiangxiang Chu, Xiaoxing Wang, Bo Zhang, Shun Lu, Xiaolin Wei, and Junchi Yan. DARTS-: Robustly stepping out of performance collapse without indicators. arXiv:2009.01027, 2020.

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

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

[43] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott E. Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proc. CVPR*, pages 1–9, 2015.

[44] Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. MobileNets: Efficient convolutional neural networks for mobile vision applications. arXiv:1704.04861, 2017.

[45] Ningning Ma, Xiangyu Zhang, Hai-Tao Zheng, and Jian Sun. ShuffleNet V2: Practical guidelines for efficient CNN architecture design. In *Proc. ECCV*, pages 122–138, 2018.

[46] Mingxing Tan, Bo Chen, Ruoming Pang, Vijay Vasudevan, Mark Sandler, Andrew Howard, and Quoc V. Le. MnasNet: Platform-aware neural architecture search for mobile. In *Proc. CVPR*, pages 2820–2828, 2019.

[47] Han Cai, Ligeng Zhu, and Song Han. ProxylessNAS: Direct neural architecture search on target task and hardware. In *Proc. ICLR*, 2019.

[48] Yann LeCun, Léon Bottou, Genevieve B. Orr, and Klaus-Robert Müller. Efficient backprop. In *Neural Networks: Tricks of the Trade (2nd ed.)*, Lecture Notes in Computer Science, pages 9–48. 2012.

[49] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *AISTATS*, volume 9 of *JMLR Proceedings*, pages 249–256, 2010.

[50] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proc. ICCV*, pages 1026–1034, 2015.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] See Sec 3.1.
   (c) Did you discuss any potential negative societal impacts of your work? [No] I do not see any potential negative societal impacts of this paper.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes]
   (b) Did you include complete proofs of all theoretical results? [Yes]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes]
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes]
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes]
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [No] All assets I have used are public.
   (c) Did you include any new assets either in the supplemental material or as a URL? [N/A]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]