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

Yao Shu[†], **Zhongxiang Dai**[†], **Zhaoxuan Wu**[§], **Bryan Kian Hsiang Low**[†]
Dept. of Computer Science, National University of Singapore, Republic of Singapore[†]
Institute of Data Science, National University of Singapore, Republic of Singapore[§]
Integrative Sciences and Engineering Programme, NUSGS, Republic of Singapore[§]
{shuyao,daizhongxiang,lowkh}@comp.nus.edu.sg[†]
wu.zhaoxuan@u.nus.edu[§]

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., the superior search efficiency) and training-based (i.e., the remarkable search effectiveness) NAS, which we have demonstrated through extensive experiments. Our codes are available at https://github.com/shuyao95/HNAS.

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. 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 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].

Correspondence to: Zhongxiang Dai <daizhongxiang@comp.nus.edu.sg>.

36th Conference on Neural Information Processing Systems (NeurIPS 2022).

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, 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 unified generalization bound for DNNs with these metrics and then use it 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 correlation 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 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 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 $f(x; \theta)$

be the output of a DNN with input $x \in \mathbb{R}^{n_0}$ and parameters $\theta \in \mathbb{R}^d$ that are initialized using the standard normal distribution, the NTK matrix $\Theta \in \mathbb{R}^{m \times m}$ over a dataset of size m is defined as

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

Jacot et al. [13] have shown that this NTK matrix will finally converge to a deterministic form Θ_1 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., Θ_1, 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., Θ_0, based on the initialized model parameters θ_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 are expected to be applicable to the NAS search spaces with complex architectures, which we will 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 $S = \{(x_i, y_i)\}_{i=1}^m$ as

$$M_{\text{Grad}} = \frac{1}{m} \sum_{i=1}^{m} \left\| \nabla_\theta \ell(f(x_i; \theta_0), y_i) \right\|_2. \tag{2}$$

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 $H_i \in \mathbb{R}^{d \times d}$ denote the hessian matrix induced by input x_i, the metrics of SNIP and GraSP on dataset $S = \{(x_i, y_i)\}_{i=1}^m$ can be defined as

$$M_{\text{SNIP}} = \frac{1}{m} \sum_{i} \left| \theta_0 \nabla_\theta \ell(f(x_i; \theta_0), y_i) \right|; \quad M_{\text{GraSP}} = \frac{1}{m} \sum_{i} \theta_0 (H_i \nabla_\theta \ell(f(x_i; \theta_0), y_i)). \tag{3}$$

Of note, we use the scaled (i.e., by m) absolute value of the original GraSP metric in [11] throughout this paper to match the mathematical form of other training-free metrics.

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. In addition, 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 θ_0. Let Θ_0 be the NTK matrix based on initialized model parameters θ_0 of a DNN, then without considering the constraint in [8], we frame this training-free metric on dataset $S = \{(x_i, y_i)\}_{i=1}^m$ as

$$M_{\text{Trace}} = \sqrt{k_{tr}(\Theta_0)/m}. \tag{4}$$

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 neural architectures. We validate this hypothesis a rmatively and use the following theorem to establish the theoretical connections among these metrics.

Theorem 1. Let the loss function $\grave{}(\ ;\ )$ in gradient-based training-free metrics be L-Lipschitz continuous and $\ $-Lipschitz smooth in the rst argument. There exist the constants $C_1, C_2, C_3 > 0$ such that the following holds with a high probability,

$$M_{\text{Grad}} \quad C_1 M_{\text{Trace}}; \; M_{\text{SNIP}} \quad C_2 M_{\text{Trace}}; \; M_{\text{GraSP}} \quad C_3 M_{\text{Trace}}:$$

The proof of Theorem 1 are given in Appendix A.1. Notably, our Theorem 1 implies that with a high probability, architectures of larger M_{Grad}, M_{SNIP} or M_{Grad} will also achieve a larger M_{Trace} given the inequalities above. That is, the values of M_{Grad}, M_{SNIP} and M_{Grad} for different architectures in the NAS search space should be highly correlated with the value of M_{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 M_{Trace} despite their distinct mathematical forms. Note that though our Theorem 1 is only able to establish the theoretical connections between M_{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 also serve as principled justi cations for the similar performances achieved by these training-free metrics in [9].

4.2 A Generalization Bound Induced by Training-free Metrics

Let dataset $S = \{(x_i, y_i)\}_{i=1}^m$ be randomly sampled from a data distribution D, we denote $L_S(\ )$ as the training error on S and $L_D(\ )$ as the corresponding generalization error on 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 uni ed generalization analysis for DNNs in terms of these metrics by making use of the NTK theory [13]. De ne $\grave{}(f, y) \ , \ (f \quad y)^2/2$ and $_0 \ , \ \min\{2n^{-1}(_{\min}(\ _1) + _{\max}(\ _1))^{-1}; m_{\max}^{-1}(\ _0)\}$ with $_{\min}(\ ); _{\max}(\ )$ being the minimum and maximum eigenvalue of a matrix respectively, we derive the following theorem:

Theorem 2. Assume $\|x_i\|_2 \quad 1$ and $|f(x_i; \ _0)|, \ _{\min}(\ _0), |y_i| \ 2 \ [0, 1]$ for any $(x_i, y_i) \ 2 \ S$. There exists a constant $N \ 2 \ N$ such that for any $n > N$, when applying gradient descent with learning rate $< \ _0$, the generalization error of f_t at time $t > 0$ can be bounded as below with a high probability,

$$L_D(f_t) \quad L_S(f_t) + O(\ = M):$$

Here, M can be any metric in Sec. 3.2 and $\ _{\max}(\ _0)/\ _{\min}(\ _0)$ is the condition number of $\ _0$.

Its proof is in Appendix A.2 and the second term $O(\ = M)$ 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 $L_S(\ )$ in our Theorem 2 may also depend on the training-free metric M, it also needs to be taken into account when analyzing the generalization performance (or the generalization error $L_D(\ )$) 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 $L_S(\ )$ in Theorem 2, which nally give rise to principled interpretations for different training-free NAS methods [7–9].

The realizable scenario. Similar to [18], we assume that a zero training error (i.e., $L_S(f_t) \to 0$ 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 M from Sec. 3.2, the following holds with a high probability,

$$L_D(f_t) \propto O(1/M).$$

Corollary 1 is obtained by introducing $L_S(f_t) = 0$ 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 M generally achieves an improved generalization performance. This implies that in order to select well-performing architectures, we can simply maximize M to find $A^* = \arg\max_A M(A)$ where 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 M_{Grad}, M_{SNIP} and M_{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 [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 M from Sec. 3.2 in the non-realizable scenario, there exists a constant $C > 0$ such that with a high probability,

$$L_D(f_t) \propto \frac{1}{2}\, m \quad M^2/C^{2t} + O(1/M).$$

Its proof is given in Appendix A.3. Notably, our Corollary 2 suggests that when $M \in [0, \sqrt{mC/\ln 2}]$, an architecture with a larger value of the metric 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 M_{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 M between the model convergence (i.e., the first term) and the generalization gap (i.e., the second term) when $M > \sqrt{mC/\ln 2}$, 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 M, the ones with smaller condition numbers 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 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), to further support the practicality and the superiority of our Corollary 2. In particular, inspired by the theoretical 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:

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

where $W^{(i)} \in \mathbb{R}^{n \times n}$ for any $i \in \{1, \ldots, L\}$ and every element of $W^{(i)}$ is independently initialized using the standard normal distribution. Here, $\mathbf{1}_m$ denotes an m-dimensional vector with every element being one. Let Θ_0 and Θ_0' be the NTK matrices of f and f' that are evaluated on the finite dataset $S = \{(x_i, y_i)\}_{i=1}^m$, respectively, we derive the following theorem:

Theorem 3. Let dataset S be normalized using its statistical mean and covariance such that $\mathbb{E}[x] = 0$ and $X^\top X = I$ given $X = [x_1 x_2 \cdots x_m]$, we have

$$\Theta_0 = Ln \cdot I, \quad \mathbb{E}\left[\frac{\Theta_0'}{n}\right] = Ln^L \cdot I.$$

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 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 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 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, thus further implying that our Corollary 2 is more practical and superior to our Corollary 1. More interestingly, based on the definition of M_{Trace} (4), Theorem 3 also indicates that deep architectures are expected to have larger values of M_{Trace} (due to the larger scale of $\mathbb{E}[\frac{\Theta_0'}{n}]$ 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 A be any architecture in the search space and M be any training-free metric from Sec. 3.2, then NAS problem can be formulated below in a unified manner:

$$\min_A \frac{1}{2} m \quad M^2(A)/C^{2t} + O(\kappa(A)/M(A)). \tag{6}$$

We further reformulate (6) into the following form:

$$\min_A \kappa(A)/M(A) + F(M^2(A)), \tag{7}$$

where $F(x) \uparrow x^{2t}$, and α and β are hyperparameters we introduced to absorb the impact of all other parameters in (6). Compared with the diverse form of NAS objectives in [7–9], our (7) presents a non-trivial unified form of NAS objectives for all the training-free metrics from Sec. 3.2, making it easier for practitioners to deploy NAS with different types of evaluated training-free metrics. 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 heuristic objective in [7]. Moreover, our (7) advances the training-free NAS method based on M_{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 gradient-based 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 (7) is expected to be able to lead to improved performances over other existing training-free NAS methods.

5.2 Optimization and Search Algorithm

Our theoretically motivated NAS objective (7) has unified all training-free metrics from Sec. 3.2 and improved over existing training-free NAS methods. However, its practical deployment requires

Algorithm 1: Hybrid Neural Architecture Search (HNAS)

1: Input: Training and validation dataset, metric $\mathcal{M}$ evaluated on architecture pool $\mathcal{P}$, $F(\cdot)$ for (7), evaluation history $\mathcal{H}_0 = \emptyset$, a BO algorithm $\mathcal{B}$, number of iterations/queries K
2: **for** iteration $k = 1, \ldots, K$ **do**
3: Choose α_k, β_k using the BO algorithm $\mathcal{B}$
4: Obtain the optimal candidate A_k in $\mathcal{P}$ by solving (7)
5: Evaluate the validation performance of A_k, e.g., $L_{val}(A_k)$ after training A_k
6: Update the GP surrogate in the BO algorithm $\mathcal{B}$ using the evaluation history $\mathcal{H}_k = \mathcal{H}_{k-1} \cup \{((\alpha_k, \beta_k), L_{val}(A_k))\}$
7: **end for**
8: Select the final A^* with the best validation performance, e.g., $A^* = \arg\min_{A \in \{A_k\}_{k=1}^K} L_{val}(A)$

(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 is given in the corner of each plot.

the determination of the hyperparameters α and β,[1] which can be non-trivial in practice. To this end, we further introduce Bayesian optimization (BO) [20] to optimize the hyperparameters α and β in order to maximize the true validation performance of the architectures selected by different α and β. In particular, BO uses a Gaussian process (GP) as a surrogate to model the objective function (i.e., the validation performance here) in order to sequentially choose the queried inputs (i.e., the values of α and β). This finally completes our theoretically grounded NAS framework called Hybrid NAS (HNAS), which not only novelly unifies all training-free metrics from Sec. 3.2 but also boosts NAS algorithms based on these training-free metrics in a principled way (Algorithm 1).

Specifically, in every iteration k of HNAS, we first select the optimal candidate A_k by maximizing our training-free NAS objective in (7) using the values of α and β queried by the BO algorithm in the current iteration (line 3-4 of Algorithm 1). Next, we evaluate the validation performance of A_k (e.g., validation error $L_{val}(A_k)$) and then use it to update the GP surrogate that is applied in the BO algorithm (line 5-6 of Algorithm 1), which then will be used to choose the values of α and β 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, $A^* = \arg\min_{A \in \{A_k\}_{k=1}^K} L_{val}(A)$ (see Appendix B.1 for more optimization details of Algorithm 1). Thanks to the utilization 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., the superior search efficiency) and training-based NAS (i.e., the remarkable search effectiveness) as supported by our extensive empirical results in Sec. 6.4. In addition, by novelly introducing BO to optimize the low-dimensional continuous hyperparameters α and β rather than the high-dimensional discrete architectural hyperparameters in the NAS search space, HNAS is able to avoid the issues of high-dimensional discrete optimization that standard BO algorithms usually attain when they are directly applied for NAS [21], allowing HNAS to be more efficient and effective in practice as empirically supported in our Sec. 6.4.

6 Experiments

6.1 Connections among Training-Free Metrics

We first 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 [22] and

[1] Of note, we usually fix $\alpha = 1$, which is already reasonably good for $F(\cdot)$. So, the practical deployment of (7) will mainly be affected by the choice of β.

Table 1: Correlation coefficients 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				
M_{Grad}	0.25	0.17	0.64	0.47
M_{SNIP}	0.21	0.15	0.64	0.47
M_{GraSP}	0.45	0.31	0.57	0.40
M_{Trace}	0.30	0.21	0.54	0.39
Non-realizable scenario				
M_{Grad}	0.35	0.23	0.75	0.56
M_{SNIP}	0.37	0.25	0.75	0.56
M_{GraSP}	0.46	0.32	0.69	0.50
M_{Trace}	0.33	0.23	0.70	0.51

Table 2: Comparison of topology, M_{Trace} & 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		M_{Trace}			
	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

NAS-Bench-201 [23] with CIFAR-10 [24]. 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]. Of note, we will follow the same approach to evaluate these training-free metrics in our following sections. The results in Figure 1 show that M_{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 actually align with the interpretation of our Theorem 1 (Sec. 4.1). Moreover, the correlation between any two training-free metrics from Sec. 3.2 is in Appendix C.1, which further verifies the connection among all these training-free metrics. Above all, in addition to the theoretical justification in our Theorem 1, our empirical results have also supported the connections among all the training-free metrics from Sec. 3.2.

6.2 Generalization Guarantees for Training-Free NAS

We then demonstrate 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 employ BO with a sufficiently large number of iterations (e.g., hundreds of iterations) to determine the non-trivial parameters in Corollary 2. Table 1 summarizes the results on CIFAR-10 where a higher positive correlation implies a better agreement between our generalization guarantee and the generalization performance of architectures. Notably, the generalization bound in the realizable scenario performs a compelling characterization of the test errors in NAS-Bench-201 with relatively high positive correlations, whereas it fails to provide a precise characterization in a larger search space, i.e., NAS-Bench-101. Remarkably, our generalization bound in the non-realizable scenario is able to perform consistent improvement over it by obtaining higher positive correlations. These results imply that the Corollary 1 may only provide a good characterization for training-free NAS in certain cases (e.g., in the small-scale search space NAS-Bench-201), whereas our Corollary 2 generally is more valid and robust in practice. As a consequence, following Corollary 2 should be able to improve over the NAS objective following Corollary 1 as we have justified in Sec. 5. Interestingly, the comparable results achieved by all training-free metrics from Sec. 3.2 again validate the connections among these metrics (Theorem 1). Moreover, our additional 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 support the theoretical connections between architecture topology (wide vs. deep) and the value of training-free metric M_{Trace} as well as the condition number shown in Sec. 4.4, we compare the topology width/depth, M_{Trace} and of the architectures selected by different SOTA training-based NAS algorithms in the DARTS search space, including NASNet [25], AmoebaNet [26], ENAS [4], DARTS [5], and SNAS [27]. Table 2 summarizes the results where we apply the same definition of topology width/depth in [19] (refer to [19] for more details). We also include the widest (called WIDE) and the deepest (called DEEP) architecture in the DARTS search space into this comparison. As shown in our Table 2, wide architectures (i.e., all architectures except DEEP) consistently achieve lower condition number and smaller values of M_{Trace} than deep architecture (i.e., DEEP), which aligns with our theoretical insights in Sec. 4.4.

Table 3: Comparison of 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.

Algorithm	Test Accuracy (%)			Cost (GPU Sec.)	Method	Applicable Space
	C10	C100	IN-16			
ResNet [28]	93.97	70.86	43.63	-	manual	-
REA[y]	93.92 0.30	71.84 0.99	45.15 0.89	12000	evolution	C & D
RS (w/o sharing)[y]	93.70 0.36	71.04 1.07	44.57 1.25	12000	random	C & D
REINFORCE[y]	93.85 0.37	71.71 1.09	45.24 1.18	12000	RL	C & D
BOHB[y]	93.61 0.52	70.85 1.28	44.42 1.49	12000	BO+bandit	C & D
ENAS[z] [4]	93.76 0.00	71.11 0.00	41.44 0.00	15120	RL	C
DARTS (1st)[y] [5]	54.30 0.00	15.61 0.00	16.32 0.00	16281	gradient	C
DARTS (2nd)[y] [5]	54.30 0.00	15.61 0.00	16.32 0.00	43277	gradient	C
GDAS[z] [29]	93.44 0.06	70.61 0.21	42.23 0.25	8640	gradient	C
DrNAS[♭] [30]	93.98 0.58	72.31 1.70	44.02 3.24	14887	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
GradSign [31]	93.31 0.47	70.33 1.28	42.42 2.81	-	training-free	C & D
HNAS (M_{Grad})	94.04 0.21	71.75 1.04	45.91 0.88	3010	hybrid	C & D
HNAS (M_{SNIP})	93.94 0.02	71.49 0.11	46.07 0.14	2976	hybrid	C & D
HNAS (M_{GraSP})	94.13 0.13	72.59 0.82	46.24 0.38	3148	hybrid	C & D
HNAS (M_{Trace})	94.07 0.10	72.30 0.70	45.93 0.37	3006	hybrid	C & D
Optimal	94.37	73.51	47.31	-	-	-

[y] Reported by Dong and Yang [23].
[z] Re-evaluated using the codes provided by Dong and Yang [23].
[♭] Re-evaluated under a comparable search budget as other training-based NAS algorithms with first-order optimization, e.g., ENAS and DARTS (1st). Note that this search budget is smaller than the one reported in its original paper and hence will lead to decreased search performances.

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). We refer to Appendix B.2 for our experimental details. As summarized in 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 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.5 further includes the impressive search results achieved by HNAS on CIFAR-10/100 and ImageNet in the DARTS search space. Overall, our 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.

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 to provide valuable prior knowledge for the design of training-free

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 $\mathcal{M}_{\text{Trace}}$. Note that each algorithm is reported with the mean and standard error of ten independent searches, and the black dashed line in each plot denotes the the minimal (optimal) test error that can be achieved by the architectures in NAS-Bench-201 on the corresponding dataset.

metrics and NAS search space in the future. Moreover, we expect our theoretical analyses for DNNs to be capable of inspiring more theoretical understanding and improvement over existing machine learning algorithms that are based on DNNs, e.g., the recent training-free data valuation algorithm [32]. In addition, 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 fields in order to enjoy the advantages of these two types of methods simultaneously.

Acknowledgments and Disclosure of Funding

This research/project is supported by the National Research Foundation Singapore and DSO National Laboratories under the AI Singapore Programme (AISG Award No: AISG2-RP-2020-018) and by A*STAR under its RIE2020 Advanced Manufacturing and Engineering (AME) Programmatic Funds (Award A20H6b0151).

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, 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, 2021.

[13] Arthur Jacot, Clément Hongler, and Franck Gabriel. Neural Tangent Kernel: Convergence and generalization in neural networks. In Proc. NeurIPS, 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, 2019.

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

[17] Greg Yang and Etai Littwin. Tensor programs IIb: Architectural universality of neural tangent kernel training dynamics. In Proc. ICML, 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. NIPS, 2012.

[21] Han Shi, Renjie Pi, Hang Xu, Zhenguo Li, James Kwok, and Tong Zhang. Bridging the gap between sample-based and one-shot neural architecture search with BONAS. In Proc. NeurIPS, 2020.

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

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

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

[25] Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V. Le. Learning transferable architectures for scalable image recognition. In Proc. CVPR, 2018.

[26] Esteban Real, Alok Aggarwal, Yanping Huang, and Quoc V. Le. Regularized evolution for image classifier architecture search. In Proc. AAAI, 2019.

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

[28] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In Proc. CVPR, 2016.

[29] Xuanyi Dong and Yi Yang. Searching for a robust neural architecture in four GPU hours. In Proc. CVPR, 2019.

[30] Xiangning Chen, Ruochen Wang, Minhao Cheng, Xiaocheng Tang, and Cho-Jui Hsieh. DrNAS: Dirichlet neural architecture search. In Proc. ICLR, 2021.

[31] Zhihao Zhang and Zhihao Jia. Gradsign: Model performance inference with theoretical insights. In Proc. ICLR, 2022.

[32] Zhaoxuan Wu, Yao Shu, and Bryan Kian Hsiang Low. DAVINZ: Data valuation using deep neural networks at initialization. In Proc. ICML, 2022.

[33] Beatrice Laurent and Pascal Massart. Adaptive estimation of a quadratic functional by model selection. Annals of Statistics, pages 1302–1338, 2000.

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

[35] Zhongxiang Dai, Yao Shu, and Bryan Kian Hsiang Low. Sample-then-optimize batch neural thompson sampling. In Proc. NeurIPS, 2022.

[36] Sebastian Shenghong Tay, Chuan Sheng Foo, Daisuke Urano, Richalynn Chiu Xian Leong, and Bryan Kian Hsiang Low. Efficient distributionally robust Bayesian optimization with worst-case sensitivity. In Proc. ICML, 2022.

[37] Zhongxiang Dai, Yizhou Chen, Haibin Yu, Bryan Kian Hsiang Low, and Patrick Jaillet. On provably robust meta-Bayesian optimization. In Proc. UAI, 2022.

[38] Quoc Phong Nguyen, Sebastian Tay, Bryan Kian Hsiang Low, and Patrick Jaillet. Top-k ranking bayesian optimization. In Proc. AAAI, 2021.

[39] Quoc Phong Nguyen, Zhaoxuan Wu, Bryan Kian Hsiang Low, and Patrick Jaillet. Trusted-maximizers entropy search for efficient bayesian optimization. In Proc. UAI, 2021.

[40] Zhongxiang Dai, Bryan Kian Hsiang Low, and Patrick Jaillet. Federated Bayesian optimization via Thompson sampling. In Proc. NeurIPS, 2020.

[41] Zhongxiang Dai, Bryan Kian Hsiang Low, and Patrick Jaillet. Differentially private federated Bayesian optimization with distributed exploration. In Proc. NeurIPS, 2021.

[42] Sreejith Balakrishnan, Quoc Phong Nguyen, Bryan Kian Hsiang Low, and Harold Soh. Efficient exploration of reward functions in inverse reinforcement learning via Bayesian optimization. In Proc. NeurIPS, 2020.

[43] Rachael Hwee Ling Sim, Yehong Zhang, Bryan Kian Hsiang Low, and Patrick Jaillet. Collaborative Bayesian optimization with fair regret. In Proc. ICML, 2021.

[44] Yehong Zhang, Zhongxiang Dai, and Bryan Kian Hsiang Low. Bayesian optimization with binary auxiliary information. In Proc. UAI, 2019.

[45] Zhongxiang Dai, Haibin Yu, Bryan Kian Hsiang Low, and Patrick Jaillet. Bayesian optimization meets Bayesian optimal stopping. In Proc. ICML, 2019.

[46] Zhongxiang Dai, Yizhou Chen, Bryan Kian Hsiang Low, Patrick Jaillet, and Teck-Hua Ho. R2-B2: recursive reasoning-based Bayesian optimization for no-regret learning in games. In Proc. ICML, 2020.

[47] Arun Verma, Zhongxiang Dai, and Bryan Kian Hsiang Low. Bayesian optimization under stochastic delayed feedback. In Proc. ICML, 2022.

[48] Niranjan Srinivas, Andreas Krause, Sham M. Kakade, and Matthias W. Seeger. Gaussian process optimization in the bandit setting: No regret and experimental design. In Proc. ICML, 2010.

[49] Donald R Jones, Matthias Schonlau, and William J Welch. Efficient global optimization of expensive black-box functions. Journal of Global optimization, 13(4):455–492, 1998.

[50] Fernando Nogueira. Bayesian Optimization: Open source constrained global optimization tool for Python, 2014–. URL https://github.com/fmfn/BayesianOptimization .

[51] 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.

[52] 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.

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

[54] 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.

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

[56] Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Q. Weinberger. Densely connected convolutional networks. In Proc. CVPR, 2017.

[57] 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, 2018.

[58] Renqian Luo, Fei Tian, Tao Qin, Enhong Chen, and Tie-Yan Liu. Neural architecture optimization. In Proc. NeurIPS, 2018.

[59] Quanming Yao, Ju Xu, Wei-Wei Tu, and Zhanxing Zhu. Efficient neural architecture search via proximal iterations. In Proc. AAAI, 2020.

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

[61] 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.

[62] Xiangning Chen and Cho-Jui Hsieh. Stabilizing differentiable architecture search via perturbation-based regularization. In Proc. ICML, 2020.

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

[64] 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, 2015.

[65] 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.

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

[67] 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, 2019.

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

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

[70] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In Proc. AISTATS, 2010.

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

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately re ect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] See Sec 3.1, i.e., the simpli ed analysis on the fully connected neural networks with scalar output.
   (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 identi able 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]

Appendix A Proofs

Throughout the proofs of this paper, we use lower-case bold-faced symbols to denote column vectors (e.g., $\mathbf{x}$), and upper-case bold-faced symbols to represent matrices (e.g., $\mathbf{A}$).

A.1 Proof of Theorem 1

Connecting M_{Grad} with M_{Trace}. As the loss function $\ell(\cdot\,;\cdot)$ is assumed to be ρ-Lipschitz continuous in the first argument, the following holds based on the notations in Sec. 3:

$$
\begin{aligned}
M_{Grad} &\overset{(a)}{=} \frac{1}{m}\sum_{i=1}^{m} \Big\| \nabla_{\theta}\ell(f(\mathbf{x}_i;\theta_0),y_i) \Big\|_2^2 \\
&\overset{(b)}{\geq} \frac{1}{m}\sum_{i=1}^{m} \big\| \nabla_{\theta}\ell(f(\mathbf{x}_i;\theta_0),y_i)\big\|_2 \\
&\overset{(c)}{\geq} \frac{1}{m}\sum_{i=1}^{m} \frac{1}{\rho}\big\| \nabla_{\theta} f(\mathbf{x}_i;\theta_0)\big\|_2 \\
&\overset{(d)}{\geq} \frac{1}{m}\sum_{i=1}^{m} \big\| \nabla_{\theta} f(\mathbf{x}_i;\theta_0)\big\|_2 \\
&\overset{(e)}{\geq} \frac{1}{m}\sqrt{m\sum_{i=1}^{m} \big\| \nabla_{\theta} f(\mathbf{x}_i;\theta_0)\big\|_2^2} \\
&\overset{(f)}{=} M_{Trace}
\end{aligned}
\tag{8}
$$

where we let $\nabla_{\theta} f(\mathbf{x}_i;\theta_0)$ be the gradient of the output of DNN model. Note that (a) follows from the definition of M_{Grad} in Sec. 3.2 and (b) derives from the Minkowski inequality. In addition, (d) is from the definition of Lipschitz continuity and (e) follows from the Cauchy-Schwarz inequality. Finally, (f) is based on the definition of NTK matrix in Sec. 3.1 and M_{Trace} in Sec. 3.2, i.e.,

$$
M_{Trace} = \sqrt{\frac{1}{m}\|\mathbf{\Theta}_0\|_{tr}} = \sqrt{\frac{1}{m}\sum_{i=1}^{m} \big\| \nabla_{\theta} f(\mathbf{x}_i;\theta_0)\big\|_2^2}. \tag{9}
$$

Let $C_1 = \rho$, we then have

$$
M_{Grad} \geq C_1 M_{Trace}. \tag{10}
$$

Connecting M_{SNIP} with M_{Grad}. We firstly introduce the following lemma.

Lemma A.1 (Laurent and Massart [33]). *If $x_1, \ldots, x_k$ are independent standard normal random variables, for $y = \sum_{i=1}^{k} x_i^2$ and any $t > 0$,*

$$
P(y - k \geq 2\sqrt{k}t + 2t) \leq \exp(-t).
$$

Following the common practice in [13, 14], each element of $\theta_0 \in \mathbb{R}^d$ follows from the standard normal distribution independently. We therefore can bound $\|\mathbf{\Theta}_0\|_2^2$ using the lemma above. Specifically, let $t = \exp(\epsilon) \in (0, 1)$, with probability at least $1-\delta$ over random initialization, we have:

$$
\|\mathbf{\Theta}_0\|_2^2 \leq d + 2\sqrt{d\ln\frac{1}{\delta}} + 2\ln\frac{1}{\delta}. \tag{11}
$$

Using the results above and following the de nition of M_{Grad}, with probability at least 1 over random initialization, we have

$$\begin{aligned} M_{SNIP} &= \frac{1}{m}\sum_{i=1}^{m} \geq_0 r \quad L(f(x_i;\ _0); y_i) \\ &\quad \frac{1}{m}\sum_{i=1}^{m} k\ _0 k_2 k r \quad `(f(x_i;\ _0); y_i)k_2 \\ &\quad s\ \frac{r}{d+2\quad d\ln\frac{1}{-}+2\ln\frac{1}{-}}\ \frac{1}{m}\sum_{i=1}^{m} k r \quad `(f(x_i;\ _0); y_i)k_2 \\ &\quad s\ \frac{r}{d+2\quad d\ln\frac{1}{-}+2\ln\frac{1}{-}} M_{Trace}: \end{aligned} \tag{12}$$

The last inequality follows from the same derivation of (8). Let C_2 , $\frac{r}{d+2\quad d\ln\frac{1}{-}+2\ln\frac{1}{-}}$, the following then holds with a high probability (i.e., at least 1),

$$M_{SNIP} \quad C_2 M_{Trace}: \tag{13}$$

Connecting M_{GraSP} and M_{Grad}. We rstly introduce the following lemma adapted from [16].

Lemma A.2 (Lemma 1 in [16]). Let $2 (0; 1)$. There exist the constant $_1; _2 > 0$ such that for any $r > 0$, $; {}^0 2 B(_0; r=\sqrt{n})$ and any input x within the dataset, with probability at least 1 over random initialization, we have

$$\begin{aligned} & k r \quad f(x; \ ) k_2 \quad {}_1 \\ k r \quad f(x; \ ) r \quad {}_0 f(x; \ {}^0) k_2 \quad & {}_2 k \quad {}^0 k_2 \end{aligned}$$

where $B(_0; r=\sqrt{n})$, $f \quad j k \quad {}_0 k \quad r=\sqrt{n} g$.

To ease the notation, we use $r_f `(f(x_i;\ _0); y_i)$ to denote the gradient of the output (i.e., $f(x_i;\ _0)$) from the DNN model f . According to the de nition of Hessian matrix, H_i applied in M_{GraSP} can be computed as

$$\begin{aligned} H_i &= r^2_{\ _0} `(f(x_i;\ _0); y_i) \\ &= r \quad [r_f `(f(x_i;\ _0); y_i) r \quad f(x_i;\ _0)] \\ &= r^2_f `(f(x_i;\ _0); y_i) r \quad f(x_i;\ _0) r \quad f(x_i;\ _0)^> + r_f `(f(x_i;\ _0); y_i) r^2 f(x_i;\ _0): \end{aligned} \tag{14}$$

Since $`(\ ;\ )$ is assumed to be $$-Lipschitz smooth and $$-Lipschitz continuous in the rst argument, we can then bound the operator norm of this hessian matrix, denoted by $kH_i k_2$ induced by the input x_i in the dataset with

$$\begin{aligned} kH_i k_2 = & \ r^2_f `(f(x_i;\ _0); y_i) r \quad f(x_i;\ _0) r \quad f(x_i;\ _0)^> + r_f `(f(x_i;\ _0); y_i) r^2 f(x_i;\ _0)\ _2 \\ & r^2_f `(f(x_i;\ _0); y_i) \quad r \quad f(x_i;\ _0) r \quad f(x_i;\ _0)^>\ _2 + j r_f `(f(x_i;\ _0); y_i) j\ r^2 f(x_i;\ _0)\ _2 \\ & r \quad f(x_i;\ _0) r \quad f(x_i;\ _0)^>\ _2 + \quad r^2 f(x_i;\ _0)\ _2 \\ = & \quad k r \quad f(x_i;\ _0) k_2^2 + \quad r^2 f(x_i;\ _0)\ _2 \\ & {}^2_1 + \quad {}_2 \end{aligned} \tag{15}$$

where the last inequality results from Lemma A.2 and is satis ed with probability at least 1 over random initialization.

Finally, let $\delta' \in (0,1)$, based on the definition of M_{GraSP}, the following then holds with probability at least $1-(m+1)\delta'$ over random initialization,

$$\begin{aligned} M_{GraSP} &= \frac{1}{m}\sum_{i=1}^{m} \mathbb{1}_{\geq 0}(Hr\ L(f(x_i;\theta_0),y_i)) \\ &\quad \frac{1}{m}\|\theta_0\|_2 \sum_{i=1}^{m} \|Hr\ L(f(x_i;\theta_0),y_i)\|_2 \\ &\quad \frac{1}{m}\|\theta_0\|_2 \sum_{i=1}^{m} \|H\|_2 \|r\ L(f(x_i;\theta_0),y_i)\|_2 \\ &\quad \sqrt{(\rho_1^2+\rho_2)\left(d+2\sqrt{d\ln\frac{1}{\delta'}}+2\ln\frac{1}{\delta'}\right)}\ \frac{1}{m}\sum_{i=1}^{m} \|r\ \ell(f(x_i;\theta_0),y_i)\|_2 \\ &\quad \sqrt{(\rho_1^2+\rho_2)\left(d+2\sqrt{d\ln\frac{1}{\delta'}}+2\ln\frac{1}{\delta'}\right)} M_{Trace}. \end{aligned} \tag{16}$$

Similarly, let $\rho = (m+1)\delta'$ and $C_3 = \sqrt{(\rho_1^2+\rho_2)\left(d+2\sqrt{d\ln\frac{m+1}{\rho}}+2\ln\frac{m+1}{\rho}\right)}$, with a high probability (i.e., at least $1-\rho$), we finally have

$$M_{GraSP} \quad C_3 M_{Trace}; \tag{17}$$

which concludes our proof.

Remark. In addition to the provable theoretical connection between M_{Trace} and other training-free metrics from Sec. 3.2, we can further reveal the connection between M_{Trace} and recently proposed training-free metric M_{KNAS} in [12]. Specifically, let the training-free metric M_{KNAS} be defined as

$$M_{KNAS}, \sqrt{\frac{1}{m^2}\sum_{i,j=1}^{m} r\ f(x_i;\theta_0)^\top r\ f(x_j;\theta_0)}. \tag{18}$$

Of note, we have adapted the original KNAS metric in [12] to match the mathematical form of other training-free metrics in Sec. 3.2. Interestingly, training-free metric M_{KNAS} is also gradient-based. As a result, we can also theoretically connect M_{KNAS} with M_{Trace} in a similar way:

$$\begin{aligned} M_{KNAS}^2 &= \frac{1}{m^2}\sum_{i,j=1}^{m} r\ f(x_i;\theta_0)^\top r\ f(x_j;\theta_0) \\ &\quad \sqrt{\frac{1}{m^2}}\sqrt{m^2\sum_{i,j=1}^{m}(r\ f(x_i;\theta_0)^\top r\ f(x_j;\theta_0))^2} \\ &= \frac{1}{m}\|\theta_0\|_F\ \frac{1}{m}\|\theta_0\|_{tr} = M_{Trace}^2 \end{aligned} \tag{19}$$

where the first inequality follows from the Cauchy-Schwarz inequality and the second equality is based on the definition of Frobenius norm. The last inequality derives from the matrix inequality $\|\cdot\|_F \leq \|\cdot\|_{tr}$ while the last equality is obtained based on the definition of M_{Trace}. Therefore, we have the following theoretical connection between M_{KNAS} and M_{Trace}, which we will validate empirically in Appendix C.1.

$$M_{KNAS} \quad M_{Trace}. \tag{20}$$

Consequently, the theoretical results and the HNAS framework in this paper are also applicable to the training-free metric M_{KNAS}. We have validated part of them empirically in Appendix C.

Remark. Note that our assumptions about the Lipschitz continuity and the Lipschitz smoothness of loss function $\ell(\cdot;\cdot)$ are usually satisfied for commonly employed loss functions in practice, e.g., Cross Entropy and Mean Square Error. For example, Shu et al. [8] have justified that these two commonly applied loss functions indeed satisfy the Lipschitz continuity assumption. As for their Lipschitz smoothness, following a similar analysis in [8], we can also verify that there exists a constant $c>0$ such that $\|\nabla_f^2 \ell(f;\cdot)\|_2 \leq c$ for both Cross Entropy and Mean Square Error.

A.2 Proof of Theorem 2

A.2.1 Estimating the Rademacher Complexity of DNNs

Note that the Rademacher complexity of a hypothesis class over dataset $S = \{(x_i, y_i)\}_{i=1}^m$ of size m is usually defined as

$$R_S(G) = \mathbb{E}_{\sigma \in \{-1,1\}^m}\left[\sup_{g \in G} \frac{1}{m} \sum_{i=1}^m \sigma_i g(x_i)\right], \tag{21}$$

with $\sigma_i \in \{-1, 1\}$. Let θ_0 be the initialized parameters of DNN model, then we define the following hypotheses that will be used to prove our lemmas and theorems:

$$F := \{x \mapsto f(x; \theta_t) : t > 0\}, \quad F^{lin} := \{x \mapsto f(x; \theta_0) + \nabla_\theta f(x; \theta_0)^\top (\theta_t - \theta_0) : t > 0\} \tag{22}$$

where $f_t \in F$ and $f_t^{lin} \in F^{lin}$ are the function determined by the DNN model and its corresponding linearization at step t of their optimization, respectively. Of note, the θ_t in f_t and f_t^{lin} are not identical and should instead be determined by the optimization of f_t and f_t^{lin} independently. Interestingly, f_t can then be well characterized by f_t^{lin} as proved in the following lemma.

Lemma A.3 (Theorem H.1 in [16]). Let $n_1 = \cdots = n_{L-1} = n$ and assume $\lambda_{min}(\Theta^{(1)}) > 0$. There exist the constant $c > 0$ and $N > 0$ such that for any $n > N$ and any $x \in \mathbb{R}^{n_0}$ with $\|x\|_2 = 1$, the following holds with probability at least $1 - \delta$ over random initialization when applying gradient descent with learning rate $< \eta_0$,

$$\sup_{t \geq 0} \left\| f_t - f_t^{lin} \right\|_2 \leq \frac{c}{\sqrt{n}}.$$

Remark. According to [16], $\lambda_{min}(\Theta^{(1)}) > 0$ usually holds especially when any input x from dataset S satisfies $\|x\|_2 = 1$. In practice, $\|x\|_2 = 1$ can be achieved by normalizing each input x from real-world dataset using its norm $\|x\|_2$, which typically servers as the data preprocessing procedure for the model training of DNNs.

Moreover, we will show that the Rademacher complexity of the DNN model during model training (i.e., F) can also be bounded using its linearization model (i.e., F^{lin}) based on the following lemmas.

Lemma A.4. With Lemma A.3, there exists a constant $c > 0$ such that with probability at least $1 - \delta$ over random initialization, the following holds

$$R_S(F) \leq R_S(F^{lin}) + \frac{c}{\sqrt{n}}.$$

Proof. Based on Lemma A.3, given $\sigma_i \in \{-1, 1\}$, with probability at least $1 - \delta$, there exists a constant $c > 0$ such that

$$\sigma_i f_t \leq \sigma_i f_t^{lin} + \frac{c}{\sqrt{n}}. \tag{23}$$

Following the definition of Rademacher complexity, we can bound the complexity of F by

$$\begin{aligned} R_S(F) &= \mathbb{E}_{\sigma \in \{-1,1\}^m}\left[\sup_{f \in F} \frac{1}{m} \sum_{i=1}^m \sigma_i f(x_i; \theta)\right] \\ &\leq \mathbb{E}_{\sigma \in \{-1,1\}^m}\left[\sup_{f^{lin} \in F^{lin}} \frac{1}{m} \sum_{i=1}^m \sigma_i f^{lin}(x_i) + \frac{c}{\sqrt{n}}\right] \\ &\leq \mathbb{E}_{\sigma \in \{-1,1\}^m}\left[\sup_{f^{lin} \in F^{lin}} \frac{1}{m} \sum_{i=1}^m \sigma_i f^{lin}(x_i)\right] + \mathbb{E}_{\sigma \in \{-1,1\}^m}\left[\frac{c}{\sqrt{n}}\right] \\ &\leq R_S(F^{lin}) + \frac{c}{\sqrt{n}}, \end{aligned} \tag{24}$$

which completes the proof. □

Lemma A.5. Let $f(X;\theta_0)$, $[f(x_1;\theta_0) \quad f(x_m;\theta_0)]^\top$ and y, $[y_1 \quad y_m]^\top$ be the outputs of DNN model f at initialization and the target labels of a dataset $S = \{(x_i, y_i)\}_{i=1}^m$, respectively. Given MSE loss $L = \sum_{i=1}^m \|f^{lin}(x_i;\theta) - y_i\|_2^2/(2m)$ and NTK matrix at initialization $\Theta_0 = \nabla_\theta f(X;\theta_0)\nabla_\theta f(X;\theta_0)^\top$, assume $\lambda_{min}(\Theta_0) > 0$, for any $t > 0$, the following holds when applying gradient descent on $f^{lin}(x;\theta)$ with learning rate $\eta < m/\lambda_{max}(\Theta_0)$:

$$\|\theta_t - \theta_0\|_2 \|\theta_1 - \theta_0\|_2 = \sqrt{y^\top \Theta_0^{-1} y}$$

where θ_t denotes the parameters of f^{lin} at step t of its model training and $y \quad f(X;\theta_0)$. Besides, $\lambda_{max}(\Theta_0)$ and $\lambda_{min}(\Theta_0)$ denote the maximum and minimum eigenvalue of matrix Θ_0.

Proof. Following the update of gradient descent on MSE with learning rate $\eta < m/\lambda_{max}(\Theta_0)$, we have

$$\theta_{t+1} = \theta_t - \frac{\eta}{m}\nabla_\theta f(X;\theta_0)^\top \left(f^{lin}(X;\theta_t) - y\right). \tag{25}$$

Note that $\nabla_\theta f(X;\theta_0)$ is a $m \quad d$ matrix and $f(X;\theta_0), f^{lin}(X;\theta_0), y$ are m-dimensional column vectors. By subtracting θ_0, multiplying $\nabla_\theta f(X;\theta_0)$ and then adding $f(X;\theta_0)$ on both sides of the equality above, we achieve

$$f(X;\theta_0) + \nabla_\theta f(X;\theta_0)(\theta_{t+1} - \theta_0) = f(X;\theta_0) + \nabla_\theta f(X;\theta_0)(\theta_t - \theta_0) - \frac{\eta}{m}\Theta_0\left(f^{lin}(X;\theta_t) - y\right), \tag{26}$$

which can be simplified as

$$\begin{aligned} f^{lin}(X;\theta_{t+1}) &= f^{lin}(X;\theta_t) - \frac{\eta}{m}\Theta_0\left[f^{lin}(X;\theta_t) - y\right] \\ &= \left(I - \frac{\eta}{m}\Theta_0\right) f^{lin}(X;\theta_t) + \frac{\eta}{m}\Theta_0 y. \end{aligned} \tag{27}$$

By recursively applying the equality above for $t+1$ times, we finally achieve

$$\begin{aligned} f^{lin}(X;\theta_{t+1}) &\overset{(a)}{=} \left(I - \frac{\eta}{m}\Theta_0\right)^{t+1} f^{lin}(X;\theta_0) + \sum_{j=0}^{t} \left(I - \frac{\eta}{m}\Theta_0\right)^j \frac{\eta}{m}\Theta_0 y \\ &\overset{(b)}{=} \left(I - \frac{\eta}{m}\Theta_0\right)^{t+1} f(X;\theta_0) + \left[I - \left(I - \frac{\eta}{m}\Theta_0\right)^{t+1}\right]\left(\frac{\eta}{m}\Theta_0\right)^{-1}\frac{\eta}{m}\Theta_0 y \\ &\overset{(c)}{=} \left(I - \frac{\eta}{m}\Theta_0\right)^{t+1}\left(f(X;\theta_0) - y\right) + y \end{aligned} \tag{28}$$

where (b) follows from the sum of geometric series for matrix with $\eta < m/\lambda_{max}(\Theta_0)$ as well as the fact that $f^{lin}(X;\theta_0) = f(X;\theta_0)$. Note that this result can be integrated into (25) and provide the following explicit form of $\theta_{t+1} \quad \theta_0$ after applying gradient descent for $t+1$ times:

$$\begin{aligned} \theta_{t+1} \quad \theta_0 &= \sum_{k=0}^{t} \theta_{k+1} \quad \theta_k \\ &= -\frac{\eta}{m}\nabla_\theta f(X;\theta_0)^\top \sum_{k=0}^{t}\left(I - \frac{\eta}{m}\Theta_0\right)^k (y \quad f(X;\theta_0)) \\ &= -\frac{\eta}{m}\nabla_\theta f(X;\theta_0)^\top \sum_{k=0}^{t}\left(I - \frac{\eta}{m}\Theta_0\right)^k y \end{aligned} \tag{29}$$

Since Θ_0 is symmetric, we can alternatively represent it as $\Theta_0 = V \Lambda V^\top$ using principal component analysis (PCA) where V and Λ denotes the matrix of eigenvectors $\{v_i\}_{i=1}^m$ and eigenvalues

$\{\lambda_i\}_{i=1}^m$, respectively. Based on this representation, we have

$$
\begin{aligned}
\|k_{t+1} - k_0\|_2 &= \sqrt{(\alpha_{t+1} - \alpha_0)^\top (\alpha_{t+1} - \alpha_0)} \\
&= \sqrt{y^\top \sum_{k=0}^{t} (I - \tfrac{1}{m} K_0)^k K \, f(X, \theta_0) - f(X, \theta_0)^\top \sum_{k'=0}^{t} (I - \tfrac{1}{m} K_0)^{k'} y} \\
&= \sqrt{y^\top \sum_{k=0}^{t} (I - \tfrac{1}{m} K_0)^k \, K_0 \sum_{k'=0}^{t} (I - \tfrac{1}{m} K_0)^{k'} y} \\
&= \sqrt{y^\top \sum_{k=0}^{t} (I - \tfrac{1}{m} V \Lambda V^\top)^k V \Lambda V^\top \sum_{k'=0}^{t} (I - \tfrac{1}{m} V \Lambda V^\top)^{k'} y} \\
&= \sqrt{y^\top V \sum_{k=0}^{t} (I - \tfrac{1}{m} \Lambda)^k V^\top V \Lambda V^\top V \sum_{k'=0}^{t} (I - \tfrac{1}{m} \Lambda)^{k'} V^\top y} \\
&= \sqrt{y^\top V \sum_{k=0}^{t} (I - \tfrac{1}{m} \Lambda)^k \sum_{k'=0}^{t} (I - \tfrac{1}{m} \Lambda)^{k'} V^\top y} \\
&= \sqrt{\sum_{i=1}^m \lambda_i \Big[\sum_{k=0}^{t} (1 - \tfrac{1}{m} \lambda_i)^k \Big]^2 (v_i^\top y)^2} \, .
\end{aligned} \tag{30}
$$

Since $\lambda_i < m = \lambda_{\max}(K_0)$ and $\lambda_{\min}(K_0) > 0$, we have $0 < 1 - \lambda_i/m < 1$ and hence

$$
\begin{aligned}
\|k_t - k_0\|_2 &= \sqrt{\sum_{i=1}^m \lambda_i \Big[\sum_{k=0}^{t} (1 - \tfrac{1}{m} \lambda_i)^k \Big]^2 (v_i^\top y)^2} \\
&< \sqrt{\sum_{i=1}^m \lambda_i \Big[\sum_{k=0}^{t+1} (1 - \tfrac{1}{m} \lambda_i)^k \Big]^2 (v_i^\top y)^2} \\
&= \|k_{t+1} - k_0\|_2
\end{aligned} \tag{31}
$$

We complete the proof by recursively applying the inequalities above

$$
\begin{aligned}
\|k_t - k_0\|_2 &< \|k_{t+1} - k_0\|_2 \\
&= \sqrt{\sum_{i=1}^m \lambda_i \Big[\sum_{k=0}^{t} (1 - \tfrac{1}{m} \lambda_i)^k \Big]^2 (v_i^\top y)^2} \\
&< \sqrt{\sum_{i=1}^m \lambda_i \Big[\frac{1}{\lambda_i / m} \Big]^2 (v_i^\top y)^2} \\
&= \sqrt{\sum_{i=1}^m \lambda_i^{-1} (v_i^\top y)^2} \\
&= \sqrt{y^\top K_0^{-1} y}
\end{aligned} \tag{32}
$$

□

Lemma A.6 (Awasthi et al. [34]). Let $G = \{x \mapsto w^T x : \|w\|_2 \le R\}$ be a family of linear functions defined over R^d with bounded weight. Then the empirical Rademacher complexity of G for m samples

$S \triangleq (x_1, \ldots, x_m)$ admits the following upper bounds:

$$R_S(\mathcal{G}) \leq \frac{R}{m}\|X^\top\|_{2,2}$$

where X is the $d \times m$ matrix with x_is as columns $X = [x_1, \ldots, x_m]$.

Based on our Lemma A.4 and Lemma A.5, we can finally bound the Rademacher complexity of a DNN model during its model training (i.e., F_t) using its linearization model (i.e., F_t^{lin}). Specifically, under the conditions in Theorem A.3 and Lemma A.5, there exist the constant $c > 0$ and $N > 0$ such that for any $n > N$, with probability at least $1 - \delta$ over initialization, we have

$$\begin{aligned}
R_S(\mathcal{F}) &\overset{(a)}{\leq} R_S(\mathcal{F}^{\text{lin}}) + \frac{c}{\sqrt{n}} \\
&\overset{(b)}{=} \mathbb{E}_{\{\epsilon_i\}_{i=1}^m}\left[\sup_{t \geq 0} \frac{1}{m}\sum_{i=1}^m \epsilon_i \left(f(x_i; \theta_0) + \nabla f(x_i; \theta_0)^\top (\theta_t - \theta_0)\right)\right] + \frac{c}{\sqrt{n}} \\
&\overset{(c)}{=} \mathbb{E}_{\{\epsilon_i\}_{i=1}^m}\left[\sup_{t \geq 0} \frac{1}{m}\sum_{i=1}^m \epsilon_i \nabla f(x_i; \theta_0)^\top (\theta_t - \theta_0)\right] + \frac{1}{m}\sum_{i=1}^m \mathbb{E}_{\{\epsilon_i\}_{i=1}^m}[\epsilon_i] f(x_i; \theta_0) + \frac{c}{\sqrt{n}} \\
&\overset{(d)}{\leq} \frac{\|\theta_1 - \theta_0\|_2 \|\nabla f(X; \theta_0)\|_{2,2}}{m} + \frac{c}{\sqrt{n}} \\
&\overset{(e)}{\leq} \frac{\|\nabla f(X; \theta_0)\|_{2,2}\sqrt{y^\top (\Theta_0)^{-1} y}}{m} + \frac{c}{\sqrt{n}} \\
&\overset{(f)}{\leq} \sqrt{\frac{2y^\top (\Theta_0)^{-1} y}{m\lambda_0}} + \frac{c}{\sqrt{n}}
\end{aligned} \tag{33}$$

where (d) derives from Lemma A.6 and (f) derives from the following inequalities based on the definition $\lambda_0 \triangleq \min(\Theta_0)/2$ and $\Theta_0 \triangleq \min(\Theta_0)$.

$$\begin{aligned}
\|\nabla f(X; \theta_0)\|_{2,2} &= \sqrt{\sum_{i=1}^m \|\nabla f(x_i; \theta_0)\|_2^2} \\
&= \sqrt{\sum_{i=1}^m \Theta_i(\theta_0)} \\
&\leq \sqrt{m\Theta_0}.
\end{aligned} \tag{34}$$

A.2.2 Deriving the Generalization Bound for DNNs using Training-free Metrics

Define the generalization error on the data distribution $\mathcal{D}$ as $L_{\mathcal{D}}(g) \triangleq \mathbb{E}_{(x,y)\sim\mathcal{D}}\ell(g(x), y)$ and the empirical error on the dataset $S = \{(x_i, y_i)\}_{i=1}^m$ that is randomly sampled from $\mathcal{D}$ as $L_S(g) \triangleq \frac{1}{m}\sum_{i=1}^m \ell(g(x_i), y_i)$. Given the loss function $\ell(\cdot, \cdot)$ and the Rademacher complexity of any hypothesis class $\mathcal{G}$, the generalization error on the hypothesis class can then be estimated by the empirical error using the following lemma.

Lemma A.7 (Mohri et al. [18]). *Suppose the loss function $\ell(\cdot, \cdot)$ is bounded in $[0, 1]$ and is ρ-Lipschitz continuous in the first argument. Then with probability at least $1 - \delta$ over dataset S of size m:*

$$\sup_{g\in\mathcal{G}}\{L_{\mathcal{D}}(g) - L_S(g)\} \leq 2\rho R_S(\mathcal{G}) + 3\sqrt{\ln(2/\delta)/(2m)}.$$

Lemma A.8. *For a symmetric matrix $A \in \mathbb{R}^{m\times m}$ with eigenvalues $\{\lambda_i\}_{i=1}^m$ in an ascending order, define $\lambda_1, \ldots, \lambda_m = \lambda_1$, the following inequality holds if $\lambda_1 > 0$,*

$$\|A\|_{\text{tr}}\|A^{-1}\|_{\text{tr}} \geq m^2.$$

Proof. Since eigenvalues $\{\lambda_i\}_{i=1}^m$ are in an ascending order, we have

$$\frac{m}{\lambda_i} \quad \lambda_1 : \tag{35}$$

Based on the results above, we can connect the matrix norms $\|A\|_{tr}$ and $\|A^{-1}\|_{tr}$ with

$$\|A\|_{tr} \quad A^{-1}\|_{tr} = \Big(\sum_{i=1}^m \quad \lambda_i\Big) \quad \Big(\sum_{i=1}^m \quad \lambda_i^{-1}\Big) \quad (m \quad \lambda_1) \quad \frac{m}{m} = \frac{m^2 \quad ^2}{} = m^2 \ ; \tag{36}$$

which concludes the proof. □

We are now able to prove Theorem 2 by combining the results in Lemma A.3 and (33). Speci cally, under the conditions in Theorem A.3 and Lemma A.5, there exist constants $c, \ln > 0$ such that for any $f_t \in \mathcal{F}$ and any $n > N$, the following holds with probability at least $1 \quad 2$ over random initialization,

$$\begin{aligned} L_D(f_t) \quad L_S(f_t) &+ 2 \quad R_S(\mathcal{F}) + 3\sqrt{\frac{\ln(2/)}{2m}} \\ L_S(f_t) &+ 2\sqrt{\frac{\mathbf{y}^\top \quad _0^{-1}\mathbf{y}}{m}} + \frac{2c}{\sqrt{n}} + 3\sqrt{\frac{\ln(2/)}{2m}} : \end{aligned} \tag{37}$$

Assume $f(x; \ _0)$ and y are bounded in $[0; 1]$ for any pair $(x; y)$ in the dataset S, let $\{\mathbf{v}_i\}_{i=1}^m$ and $\{\lambda_i\}_{i=1}^m$ be the eigenvectors and eigenvalues of H_0, respectively, we then have $\mathbf{y} \in [\ 1; 1]^m$ and the following inequalities:

$$\mathbf{y}^\top \quad _0^{-1}\mathbf{y} = \sum_{i=1}^m \frac{(\mathbf{v}_i^\top \mathbf{y})^2}{\lambda_i} \quad \sum_{i=1}^m \frac{\|\mathbf{v}_i\|_2^2\|\mathbf{y}\|_2^2}{\lambda_i} \quad \sum_{i=1}^m \frac{m}{\lambda_i} : \tag{38}$$

Based on the fact that $\|H_0\|_{tr} = \sum_{i=1}^m \lambda_i$ and Lemma A.8, we finally achieve

$$\sqrt{\frac{\mathbf{y}^\top \quad _0^{-1}\mathbf{y}}{m}} \quad \sqrt{\quad _0^{-1} \quad _{tr}} \quad \sqrt{\frac{m}{\|H_0\|_{tr}}} = \sqrt{\frac{m}{M \quad _{Trace}}} : \tag{39}$$

By introducing (39) into (37), with $_0 \quad 1$, we have

$$L_D(f_t) \quad L_S(f_t) + \frac{2\sqrt{m}}{M \quad _{Trace}} + \frac{2c}{\sqrt{n}} + 3\sqrt{\frac{\ln(2/)}{2m}} : \tag{40}$$

Let M be any metric introduced in Sec. 3.2, based on the results in our Theorem 1 and the de nition of $O(\cdot)$, the following inequality then holds with a high probability using the result above:

$$L_D(f_t) \quad L_S(f_t) + O(/M) \ ; \tag{41}$$

which finally concludes our proof of Theorem 2.

Remark. Our (41) still holds when $_0 \quad z(z \quad 1)$, i.e., by simply placing z into our (40). Though our conclusion is based on the initialization using standard normal distribution and over-parameterized DNNs, our empirical results in Appendix C.6 show that this conclusion can also hold for DNNs initialized using other methods and also DNNs of small layer width.

A.3 Proof of Corollary 2

To prove our Corollary 2, we first consider the convergence of f_t^{lin} under the same conditions in Theorem 2. Speci cally, following the notations and results in Lemma A.5, we have $\{\mathbf{v}_i\}_{i=1}^m$ and $\{\lambda_i\}_{i=1}^m$

be the eigenvectors and eigenvalues of Θ_t, respectively, we have

$$\begin{aligned} L_S(f_t^{lin}) &\overset{(a)}{=} \frac{1}{2m} \left\| f^{lin}(X;\theta_t) - y \right\|_2^2 \\ &\overset{(b)}{=} \frac{1}{2m} \left\| I - \frac{\eta}{m}\Theta_0 \right\|^t (f(X;\theta_0) - y) \Big\|_2^2 \\ &\overset{(c)}{=} \frac{1}{2m} \left\| I - \frac{\eta}{m}\Theta_0 \right\|^t y \Big\|_2^2 \\ &\overset{(d)}{=} \frac{1}{2m} \sum_{i=1}^{m} \left| 1 - \frac{\eta}{m}\lambda_i \right|^{2t} \left\| v_i^\top y \right\|_2^2 \\ &\overset{(e)}{=} \frac{1}{2m} \sum_{i=1}^{m} \left| 1 - \frac{\eta}{m}\lambda_i \right|^{2t} \left\| v_i \right\|_2^2 \left\| y \right\|_2^2 \end{aligned} \tag{42}$$

where (d) follows the same derivation in (30). Moreover, based on $y \in [-1,1]^m$ and the fact that $\|v_i\|_2 = 1$, for any $t > 0$ (i.e., $t = 1, 2, \ldots$), we naturally have

$$\begin{aligned} L_S(f_t^{lin}) &\overset{(a)}{\leq} \frac{1}{2} \sum_{i=1}^{m} \left| 1 - \frac{\eta}{m}\lambda_i \right|^{2t} \\ &\overset{(b)}{\leq} \frac{1}{2} \sum_{i=1}^{m} \left| 1 - \frac{\eta}{m}\lambda_i \right|^{2t} \\ &\overset{(c)}{=} \frac{1}{2} \left| m - \frac{\eta}{m}k_0 \right|^{2t} \\ &\overset{(d)}{=} \frac{1}{2} \left| m - \frac{\eta}{m}M_{Trace}^2 \right|^{2t} \\ &\overset{(e)}{\leq} \frac{1}{2} \left| m - \eta M^2/C \right|^{2t} \end{aligned} \tag{43}$$

where (e) is based on the results in our Theorem 1: For any training-free metric M introduced in Sec. 3.2, there exists a constant C such that the following holds with a high probability,

$$M^2 \leq C M_{Trace}^2 \quad \Rightarrow \quad m - \eta M^2/C \geq m - \eta M_{Trace}^2. \tag{44}$$

Based on Lemma A.3 and the fact that loss function $l(f, y) = (f - y)^2/2$ is 1-Lipschitz continuous in the first argument, the following then holds with a high probability

$$L_S(f_t) - L_S(f_t^{lin}) \leq \left\| f_t - f_t^{lin} \right\|_\infty \leq O\left(\sqrt{\frac{1}{n}}\right). \tag{45}$$

By introducing the results above into our Theorem 2 with $\sqrt{n}$ being absorbed in $O(\cdot)$, we finally achieve the following results with a high probability,

$$\begin{aligned} L_D(f_t) &\leq L_S(f_t) + O(\eta/M) \leq L_S(f_t^{lin}) + O(\eta/M) \\ &\leq \frac{1}{2} \left| m - \eta M^2/C \right|^{2t} + O(\eta/M), \end{aligned} \tag{46}$$

which thus concludes our proof.

A.4 Proof of Theorem 3

Let $W_j^{(i)}$ denote the j-th row of matrix $W^{(i)}$, based on the definition of f and f^0 in Sec. 4.4, we can compute the gradient (represented as a column vector) of $W_j^{(i)}$ for function f and f^0 respectively as below

$$\begin{aligned} \nabla_{W_j^{(i)}} f(x) &= x \\ \nabla_{W_j^{(i)}} f^0(x) &= \left(\prod_{k'=1}^{i-1} W^{(k')} x \right) 1^\top \left(\prod_{k=i+1}^{L} W^{(k)} \right)_j \end{aligned} \tag{47}$$

(a) Wide architecture (b) Deep architecture

Figure 3: Two different architecture topologies for our analysis.

where $\prod_{k=i+1}^{L} W^{(k)}{}_j$ is defined as the j-th column of matrix $\prod_{k=i+1}^{L} W^{(k)}$, i.e.,

$$\left(\prod_{k=i+1}^{L} W^{(k)}\right)_j = W^{(L)} W^{(L-1)} \cdots W_j^{(i+1)}; \tag{48}$$

Consequently, the NTK matrix of initialized wide architecture can be represented as

$$\begin{aligned} \Theta_0(x; x^0) &= \sum_{i=1}^{L} \sum_{j=1}^{n} \nabla_{W_j^{(i)}} f(x)^\top \nabla_{W_j^{(i)}} f(x^0) \\ &= \sum_{i=1}^{L} \sum_{j=1}^{n} x^\top x^0 = nL \, x^\top x^0. \end{aligned} \tag{49}$$

Meanwhile, the NTK matrix of initialized deep architecture can be represented as

$$\begin{aligned} \Theta_0^0(x; x^0) &= \sum_{i=1}^{L} \sum_{j=1}^{n} \nabla_{W_j^{(i)}} f^0(x)^\top \nabla_{W_j^{(i)}} f^0(x) \\ &= \sum_{i=1}^{L} \sum_{j=1}^{n} \left(\left(\prod_{k=i+1}^{L} W^{(k)} \right)_j \left(\prod_{k'=1}^{i-1} W^{(k')} x^0 \right)^{1>} \right)^\top \left(\prod_{k=i+1}^{L} W^{(k)} \right)_j \left(\prod_{k'=1}^{i-1} W^{(k')} x^0 \right)^{1>} \\ &= \sum_{i=1}^{L} \sum_{j=1}^{n} \left(1^> \left(\prod_{k=i+1}^{L} W^{(k)} \right)_j^{\top} \right) \left(\prod_{k'=1}^{i-1} W^{(k')} x^0 \right)^{>} \left(\prod_{k'=1}^{i-1} W^{(k')} x^0 \right)^{1>} \\ &= x^\top \sum_{i=1}^{L} \sum_{j=1}^{n} \left(1^> \left(\prod_{k=i+1}^{L} W^{(k)} \right)_j^{\top} \right) \left(\prod_{k'=1}^{i-1} W^{(k')} x^0 \right)^{>} \left(\prod_{k'=1}^{i-1} W^{(k')} x^0 \right)^{1>}. \end{aligned} \tag{50}$$

Since each element in $W^{(i)}$ is initialized using standard normal distribution, we have following simplified expectation by exploring the fact that $\mathbb{E}\left[W^{(i)}\right] = 0$ and $\mathbb{E}\left[W^{(i)\top} W^{(i)}\right] = nI$.

$$\begin{aligned} \mathbb{E}\left[\left(\prod_{k'=1}^{i-1} W^{(k')} \right)^{>} \left(\prod_{k'=1}^{i-1} W^{(k')} \right) \right] &= \mathbb{E}\left[W^{(1)\top} \cdots W^{(i-1)\top} W^{(i-1)} \cdots W^{(1)} \right] \\ &= \mathbb{E}\left[W^{(1)\top} \cdots \mathbb{E}\left[W^{(i-1)\top} \mathbb{E}\left[W^{(i-1)} \right] W^{(i-1)} \right] \cdots W^{(1)} \right] \\ &= \mathbb{E}\left[W^{(1)\top} \cdots \mathbb{E}\left[W^{(i-2)\top} (nI) W^{(i-2)} \right] \cdots W^{(1)} \right] \\ &= n^{i-1} I. \end{aligned} \tag{51}$$

Similarly, we also have

$$
\begin{aligned}
E\left[\left(1^{\top}\prod_{k=i+1}^{1} W^{(k)} A\right)^{\top}\right] &= 1^{\top} E\left[\prod_{k=i+1}^{1} W^{(k)} \left(\prod_{k=i+1}^{1} W^{(k)} A\right)^{\top}\right] 1 \\
&= 1^{\top} E\left[W^{(L)} E\left[\cdots E\left[W_j^{(i+1)} W_j^{(i+1)\top}\right] \cdots W^{(L)\top}\right]\right] 1 \\
&= 1^{\top} E\left[W^{(L)} E\left[\cdots E\left[W^{(i+2)} I W^{(i+2)\top}\right] \cdots W^{(L)\top}\right]\right] 1 \\
&= 1^{\top} E\left[W^{(L)} E\left[\cdots E\left[W^{(i+3)} n I W^{(i+3)\top}\right] \cdots W^{(L)\top}\right]\right] 1 \\
&= n^{L-i-1} 1^{\top} 1 \\
&= n^{L-i}.
\end{aligned}
\tag{52}
$$

Since $W^{(i)}$ in each layer is initialized independently, we achieve the following result by introducing the equality above and expectation over model parameters into (47).

$$
\begin{aligned}
E[\Theta_0(x, x')] &= x^{\top} E\left[\sum_{i=1}^{n}\sum_{j=1}^{n} \left(1^{\top}\prod_{k=i+1}^{1} W^{(k)} A\right)^{\top} \prod_{k'=1}^{i-1} W^{(k')} \prod_{k'=1}^{i-1} W^{(k')\top}\right] x' \\
&= x^{\top}\left(\sum_{i=1}^{n}\sum_{j=1}^{n} E\left[\left(1^{\top}\prod_{k=i+1}^{1} W^{(k)} A\right)^{\top}\left(\prod_{k'=1}^{i-1} W^{(k')}\right)^{\top}\left(\prod_{k'=1}^{i-1} W^{(k')}\right)\right]\right) x' \\
&= x^{\top}\left(\sum_{i=1}^{n}\sum_{j=1}^{n} n^{L-i} n^{i-1} I\right) x' \\
&= L n^{L} x^{\top} x'.
\end{aligned}
\tag{53}
$$

By exploiting the fact that $X^{\top} X = I$ with $X = [x_1, x_2, \ldots, x_m]$, we finally conclude the proof by

$$
\begin{aligned}
\Theta_0(X, X) &= L n I \\
E[\Theta_0(X, X)] &= L n^{L} I.
\end{aligned}
\tag{54}
$$

Appendix B Optimization and Experimental Details

B.1 Optimization Details for Algorithm 1

Solution to the Training-Free NAS Objective (7). Following the common practice in [6, 12], to solve (7) for the every iteration of our Algorithm 1 in practice, we independently and randomly sample a large pool of architectures from the search space to evaluate their training-free metrics and then select the architecture achieving the optimum value of (7) (given the values of θ and η) from all sampled architectures. Meanwhile, following the common practice in [9, 11], the training-free metrics of these sampled architectures are evaluated using a batch of sampled data as introduced in Sec. 6.1.

Introduction to the BO Applied in HNAS. BO is a type of gradient-free optimization algorithm aiming to optimize a black-box or non-differentiable objective function by iteratively selecting an input (to only evaluate/query its function value) that intuitively trades off between sampling an input likely achieving optimum (i.e., exploitation) given the current belief of the function modeled by a Gaussian process (GP) vs. improving the GP belief over the entire input domain (i.e., exploration) to guarantee finding the global optimum, which recently has been widely extended to various real-world problem settings in order to achieve better optimization in practice [35–47]. Since we adopt

the non-differentiable validation performance (i.e., validation error) as the objective function to be optimized (over α and β) in our Algorithm 1, BO will naturally be a better choice to find the optimal α and β compared with gradient-based optimization algorithms, and therefore has been applied in our HNAS framework. Specifically, in every iteration k of Algorithm 1, a GP belief with mean $\mu(\alpha;\beta)$ and variance $\sigma^2(\alpha;\beta)$ for the entire input domain is firstly obtained following the Equation (1) in [48] (i.e., by letting input x in [48] be the column vector $(\alpha;\beta)^\top$ and the function value y in [48] be $L_{val}(A)$) using the historical evaluations $\mathcal{H}_{k-1} = \{((\alpha_i;\beta_i), L_{val}(A_i))\}_{i=1}^{k-1}$ (this corresponds to line 6 in Algorithm 1 for iteration $k-1$).[2] Then, the mean $\mu(\alpha;\beta)$ and standard deviation $\sigma(\alpha;\beta)$ from the resulting GP belief are used to construct an acquisition function such as the expected improvement (EI) from [49] or the upper confidence bound (UCB) $\mu(\alpha;\beta) + \sqrt{\kappa}\sigma(\alpha;\beta)$ from [48] where the parameter $\kappa > 0$ is set to trade off between exploitation vs. exploration for guaranteeing no regret asymptotically with high probability. Finally, an input $(\alpha_k;\beta_k)$ will be selected (for querying) by maximizing the acquisition function within the entire input domain (i.e., line 3 in Algorithm 1), e.g., $(\alpha_k;\beta_k) = \arg\max_{(\alpha;\beta)} u(\alpha;\beta) + \sqrt{\kappa}\sigma(\alpha;\beta)$ for UCB. The acquisition function in BO is usually differentiable and thus gradient-based optimization algorithms (e.g., L-BFGS and gradient ascent) can be applied to maximize it. We refer to [48] for more technical details about the BO algorithm based on UCB and [50] for the implementation of BO that has been used in our experiments.

B.2 Experimental Details in NAS-Bench-201

In our experiments on NAS-Bench-201, we set the number of iterations K for Algorithm 1 to be 20. In addition, for every iteration of Algorithm 1, we independently and randomly sample a pool of 2,000 architectures from the search space and then choose the architecture enjoying the optimum value of (7) from all sampled architectures (e.g., $2{,}000 - k$ architectures in total). After choosing this candidate architecture, we query the validation performance of this architecture on CIFAR-10 after 12-epoch training (i.e., "hp=12") from the tabular data in NAS-Bench-201, which then will be employed to update the GP surrogate applied in BO. After completing 20 iterations of our Algorithm 1, there are (a) 40,000 sampled architectures with evaluated training-free metrics which can already cover all the architectures in NAS-Bench-201 (consisting of 15,625 architectures) with a high probability (b) and 20 architectures with evaluated validation performance which can already allow our HNAS to select architectures achieving competitive performances. Overall (c), our Algorithm 1 can be solved both efficiently and effectively following our aforementioned optimization techniques.

Appendix C More Empirical Results

C.1 Connections among Training-Free Metrics

Besides the theoretical (Theorem 1) and empirical (Sec. 4.1) connections between $\mathcal{M}_{Trace}$ and other gradient-based training-free metrics from Sec. 3.2, we further show in Table 4 that any two metrics from Sec. 3.2 are highly correlated, i.e., they consistently achieve large positive correlations in both NAS-Bench-101 and NAS-Bench-201. Similar to the results in our Sec. 4.1, the correlation between $\mathcal{M}_{GraSP}$ and any other training-free metric is generally lower than other pairs, which may result from the hessian matrix that has only been applied in $\mathcal{M}_{GraSP}$. To figure out whether our Theorem 1 is also applicable to non-gradient-based training-free metrics, we then provide the correlation between $\mathcal{M}_{Fisher}$ [51], $\mathcal{M}_{SynFlow}$ [52], $\mathcal{M}_{NASWOT}$ [6] and $\mathcal{M}_{Trace}$ [8] for the comparison. Interestingly, both $\mathcal{M}_{Fisher}$ and $\mathcal{M}_{SynFlow}$ achieve higher positive correlations with $\mathcal{M}_{Trace}$ than $\mathcal{M}_{NASWOT}$ in general. According to their mathematical forms in the corresponding papers, such a phenomenon may result from the fact that $\mathcal{M}_{Fisher}$ and $\mathcal{M}_{SynFlow}$ have contained certain gradient information while $\mathcal{M}_{NASWOT}$ only relies on the outputs of each layer in an initialized architecture.[3] These results therefore imply that our Theorem 1 may also provide valid theoretical connections for the training-free metrics that are not gradient-based but still contain certain gradient information.

[2] Since BO is usually applied to solve maximization problem, we use the historical evaluations $\mathcal{H}_{k-1} = \{((\alpha_i;\beta_i), L_{val}(A_i))\}_{i=1}^{k-1}$ for BO instead in order to maximize $-L_{val}(A)$ in practice.

[3] Of note, the so-called gradient information contained in $\mathcal{M}_{Fisher}$ and $\mathcal{M}_{SynFlow}$ is different from the commonly used gradient of initialized model parameters that is derived from loss function or the output of DNN models. So, $\mathcal{M}_{Fisher}$ and $\mathcal{M}_{SynFlow}$ are taken as the non-gradient-based training-free metrics instead in this paper.

Table 4: Connection between any two training-free metrics (M_1 and M_2 in the table) from Sec. 3.2 in NAS-Bench-101/201. Note that each training-free metric is evaluated using a batch of randomly sampled data from CIFAR-10 following that of [9].

M_1	M_2	NAS-Bench-101			NAS-Bench-201		
		Pearson	Spearman	Kendall's Tau	Pearson	Spearman	Kendall's Tau
		Gradient-based training-free metrics					
M_{Grad}	M_{SNIP}	0.98	0.98	0.87	1.00	1.00	0.97
M_{Grad}	M_{GraSP}	0.35	0.61	0.43	0.60	0.92	0.77
M_{Grad}	M_{Trace}	0.98	0.98	0.87	0.98	0.97	0.85
M_{SNIP}	M_{GraSP}	0.34	0.59	0.42	0.55	0.92	0.77
M_{SNIP}	M_{Trace}	0.94	0.93	0.77	0.97	0.96	0.83
M_{GraSP}	M_{Trace}	0.37	0.57	0.40	0.69	0.89	0.73
M_{KNAS}	M_{Grad}	0.95	0.96	0.83	0.88	0.94	0.80
M_{KNAS}	M_{SNIP}	0.91	0.92	0.75	0.87	0.94	0.78
M_{KNAS}	M_{GraSP}	0.37	0.65	0.46	0.45	0.87	0.69
M_{KNAS}	M_{Trace}	0.96	0.96	0.84	0.89	0.97	0.86
		Non-gradient-based training-free metrics					
M_{Fisher}	M_{Trace}	0.69	0.97	0.85	0.30	0.78	0.69
$M_{SynFlow}$	M_{Trace}	0.02	0.50	0.34	0.07	0.49	0.35
M_{NASWOT}	M_{Trace}	0.08	0.11	0.08	0.10	0.32	0.22

(a) Varying architecture performances in the search space

(b) Correlation between condition numbers and architecture performances

Figure 4: (a) Varying architecture performances under different value of training-free metrics in NAS-Bench-201. Note that the x-axis denotes the averaged value of training-free metrics over the architectures grouped in the same bin and y-axis denoted the test error evaluated on CIFAR-10. (b) Correlation between the condition numbers and the true generalization performances of the architectures within the same bin (i.e., the y-axis). Note that the x-axis denotes the corresponding 20 bins in Figure 4 (a).

C.2 Valid Generalization Guarantees for Training-Free NAS

To further support that our Corollary 2 presents a more practical and valid generalization guarantee for training-free NAS in practice, we examine the true generalization performances of all candidate architectures under their different value of training-free metrics in Figure 4 (a) and exhibit the correlation between the condition number and the true generalization performances of all candidate architectures in Figure 4 (b). Speci cally, we group the value of training-free metrics in NAS-Bench-201 into 20 bins and then plot the test errors on CIFAR-10 of all candidate architectures within the same bin into the blue vertical lines in Figure 4 (a). Besides, we plot the averaged test errors over the

Table 5: Correlation between the test errors of candidate architectures in NAS-Bench-201 and their training-free metrics applied in several different scenarios. We refer to Sec. 4.3 for more details about the trade-off and condition number applied in the following scenarios.

Dataset	Scenario	Spearman				Kendall's Tau			
		M_{Grad}	M_{SNIP}	M_{GraSP}	M_{Trace}	M_{Grad}	M_{SNIP}	M_{GraSP}	M_{Trace}
C10	Realizable	0.637	0.639	0.566	0.538	0.469	0.472	0.400	0.387
	Realizable + Trade-off	0.642	0.641	0.570	0.549	0.475	0.474	0.403	0.397
	Realizable + Non-realizable	0.724	0.728	0.658	0.657	0.530	0.533	0.474	0.474
		0.750	0.748	0.686	0.697	0.559	0.556	0.501	0.512
C100	Realizable	0.638	0.638	0.571	0.535	0.473	0.475	0.409	0.385
	Realizable + Trade-off	0.642	0.645	0.578	0.546	0.476	0.481	0.414	0.394
	Realizable + Non-realizable	0.716	0.719	0.649	0.651	0.527	0.529	0.469	0.470
		0.740	0.746	0.680	0.686	0.552	0.557	0.498	0.504
IN-16	Realizable	0.578	0.578	0.550	0.486	0.430	0.433	0.397	0.354
	Realizable + Trade-off	0.588	0.589	0.566	0.526	0.438	0.441	0.408	0.382
	Realizable + Non-realizable	0.646	0.649	0.612	0.587	0.472	0.474	0.443	0.423
		0.682	0.685	0.655	0.660	0.505	0.506	0.480	0.482

architectures within the same bin into the black dash lines in Figure 4 (a). Besides, each correlation between condition number and test error in Figure 4 (b) is computed using the candidate architectures within the same bin.

Notably, as illustrated by the black dash lines in Figure 4 (a), there consistently exists a trade-off for all the training-free metrics in Sec. 3.2. Specifically, there exists an optimal value M_{opt} for each training-free metric M that is capable of achieving the best generalization performance in the search space. When $M < M_{\text{opt}}$, architecture with a larger value of M typically enjoys a better generalization performance. On the contrary, when $M > M_{\text{opt}}$, architecture with a smaller value of M generally achieves a better generalization performance. Interestingly, these results perfectly align with our Corollary 2. Furthermore, Figure 4 (b) shows that the condition number is indeed highly correlated to the generalization performance of candidate architectures and a smaller condition number is generally preferred in order to select well-performing architectures in training-free NAS. More interestingly, similar phenomenons can also be found in [3, 4] and [7]. Remarkably, our Corollary 2 can provide theoretically grounded interpretations for these results, whereas Corollary 1 fails to characterize these phenomenons. Consequently, our Corollary 2 is shown to be more practical and valid in practice.

Based on the conclusions above, we then compare the impacts of the trade-off and condition number mentioned above by examining the correlation between the true generalization performances of candidate architectures and their training-free metrics applied in different scenarios. Here, we use the same parameters applied in Sec. 6.2 for Corollary 2. Table 5 summarizes the comparison. Note that the non-realizable scenario is equivalent to the realizable scenario + trade-off + as suggested by our Corollary 2. As revealed in Table 5, both trade-off and condition number are necessary to achieve an improved characterization of architecture performances over the one in the realizable scenario followed by [9], which again verifies the practicality and validity of our Corollary 2. More interestingly, condition number is shown to be more essential than the trade-off for training-free NAS in order to improve the correlations in the realizable scenario. By integrating both trade-off and condition number into the realizable scenario, the non-realizable scenario consistently enjoys the highest correlations on different datasets, which also further verifies the improvement of our training-free NAS objective (7) over the one used in [9].

C.3 Transferability of Training-Free NAS

In practice, the transferability of the architectures selected by both training-based and training-free NAS algorithms has been widely verified [5, 7, 8]. So, in this section, we also verify the transferability of our generalization guarantees for training-free NAS. Specifically, we examine the deviation of the correlation between the architecture performance and the generalization bounds in Sec. 4.3 using training-free metrics evaluated on different datasets. That is, training-free metrics and architecture performance usually will be evaluated on different datasets. Table 6 summarizes the results using CIFAR-10/100 (C10/100) and ImageNet-16-120 (IN-16) [53] in NAS-Bench-201 where we employ the same parameters as Sec. 6.2 for Corollary 2. Notably, nearly the same correlations (i.e., with

Table 6: Deviation of the correlation between the test errors in NAS-Bench-201 and the generalization bounds in Sec. 4.3 using training-free metrics evaluated on various datasets. Each correlation is reported with the mean and standard deviation using the metrics evaluated on CIFAR-10/100 and ImageNet-16-120. Small standard deviations imply strong transferability.

Dataset	Training-free Metrics			
	M_{Grad}	M_{SNIP}	M_{GraSP}	M_{Trace}
	Realizable scenario			
C10	0.64 0.01	0.64 0.01	0.58 0.02	0.55 0.01
C100	0.64 0.01	0.64 0.01	0.58 0.03	0.54 0.02
IN-16	0.57 0.01	0.57 0.01	0.52 0.03	0.47 0.02
	Non-realizable scenario			
C10	0.75 0.00	0.75 0.00	0.69 0.01	0.69 0.00
C100	0.74 0.00	0.74 0.00	0.69 0.01	0.69 0.01
IN-16	0.69 0.00	0.69 0.00	0.63 0.01	0.65 0.00

Table 7: Comparison of the number of queries (to evaluate the validation performances of trained architectures) required by different NAS algorithms in NAS-Bench-201. The performance of each algorithm is reported with the mean and standard deviation of ve independent searches.

Algorithm	Test Accuracy (%)			# Queries
	C10	C100	IN-16	
REA	93.92 0.30	71.84 0.99	45.15 0.89	102
RS (w/o sharing)	93.700.36	71.04 1.07	44.57 1.25	106
REINFORCE	93.85 0.37	71.71 1.09	45.24 1.18	103
HNAS (M_{Grad})	94.04 0.21	71.75 1.04	45.91 0.88	20
HNAS (M_{SNIP})	93.94 0.02	71.49 0.11	46.07 0.14	20
HNAS (M_{GraSP})	94.13 0.13	72.59 0.82	46.24 0.38	20
HNAS (M_{Trace})	94.07 0.10	72.30 0.70	45.93 0.37	20
Optimal	94.37	73.51	47.31	-

extremely small deviations) are achieved for training-free metrics evaluated on different datasets. This implies that the training-free metrics computed on a dataset S can also provide a good characterization of the architecture performance evaluated on another dataset S'. Therefore, the architectures selected by training-free NAS algorithms on S are also likely to produce a compelling performance on S'. That is, the transferability of the architectures selected by training-free NAS is guaranteed.

C.4 Additional Comparison in NAS-Bench-201

In addition to the comparison of search performances and search costs (measured by GPU seconds) in Table 3, we further provide the comparison of the number of queries required by different NAS algorithms in Table 7. The queries compared here are applied to evaluate the validation performance of the selected architectures after training, which is typically avoided by training-free NAS algorithms. Consequently, here, we mainly compare HNAS with other training-based NAS algorithms. As shown in Table 7, HNAS can consistently achieve improved search performances with fewer number of queries, which also aligns with the results in our Table 3. This therefore further con rms the superior search ef ciency and the remarkable search effectiveness of our HNAS framework.

C.5 HNAS in the DARTS Search Space

To support the effectiveness and ef ciency of our HNAS, we also apply HNAS in the DARTS [53] search space to nd well-performing architectures on CIFAR-10/100 and ImageNet [54]. Speci cally, we sample a pool of 60000 architecture to evaluate their training-free metrics on CIFAR-10 in order to maintain high computational ef ciency for these training-free metrics. For the results on CIFAR-10/100, we then apply the BO algorithm for 25 iterations with a 10-epoch model training

Table 8: Performance comparison among state-of-the-art (SOTA) neural architectures on CIFAR-10/100. The performance of the final architectures selected by HNAS is reported with the mean and standard deviation of five independent evaluations. The search costs are evaluated on a single Nvidia 1080Ti. Note that HNAS (C10 or C100) denoted the architecture selected by our HNAS using the dataset CIFAR-10 or CIFAR-100, respectively.

Algorithm	Test Error (%)		Params (M)		Search Cost (GPU Hours)	Search Method
	C10	C100	C10	C100		
DenseNet-BC [56]	3.46	17.18	25.6	25.6	-	manual
NASNet-A [25]	2.65	-	3.3	-	48000	RL
AmoebaNet-A [26]	3.34 ± 0.06	18.93[y]	3.2	3.1	75600	evolution
PNAS [57]	3.41 ± 0.09	19.53	3.2	3.2	5400	SMBO
ENAS [4]	2.89	19.43	4.6	4.6	12	RL
NAONet [58]	3.53	-	3.1	-	9.6	NAO
DARTS (2nd) [5]	2.76 ± 0.09	17.54[y]	3.3	3.4	24	gradient
GDAS [29]	2.93	18.38	3.4	3.4	7.2	gradient
NASP [59]	2.83 ± 0.09	-	3.3	-	2.4	gradient
P-DARTS [60]	2.50	-	3.4	-	7.2	gradient
DARTS- (avg) [61]	2.59 ± 0.08	17.51 ± 0.25	3.5	3.3	9.6	gradient
SDARTS-ADV [62]	2.61 ± 0.02	-	3.3	-	31.2	gradient
R-DARTS (L2) [63]	2.95 ± 0.21	18.01 ± 0.26	-	-	38.4	gradient
DrNAS [30]	2.46 ± 0.03	-	4.1	-	14.4	gradient
TE-NAS[♩] [7]	2.83 ± 0.06	17.42 ± 0.56	3.8	3.9	1.2	training-free
NASI-ADA [8]	2.90 ± 0.13	16.84 ± 0.40	3.7	3.8	0.24	training-free
HNAS (C10)	2.62 ± 0.04	17.10 ± 0.18	3.4	3.5	2.4	hybrid
HNAS (C100)	2.78 ± 0.05	16.29 ± 0.14	3.7	3.8	2.7	hybrid

[y] Reported by Dong and Yang [29] with their experimental settings.
Obtained by training corresponding architectures without cutout [55] augmentation.
[♩] Reported by Shu et al. [8] with their experimental settings.

for the selected architectures in our HNAS (Algorithm 1). As for the results on ImageNet, we apply the BO algorithm for 10 iterations with a 3-epoch model training for the selected architectures in our HNAS. We follow [5] to construct 20-layer final selected architectures with an auxiliary tower of weight 0.4 for CIFAR-10 (0.6 for CIFAR-100) located at 13-th layer and 36 initial channels. We evaluate these architectures on CIFAR-10/100 using stochastic gradient descent (SGD) of 600 epochs with a learning rate cosine scheduled from 0.025 to 0 for CIFAR-10 (from 0.035 to 0.001 for CIFAR-100), momentum 0.9, weight decay 3.0×10^{-4} and batch size 96. Both Cutout [55], and ScheduledDropPath linearly increased from 0 to 0.2 for CIFAR-10 (from 0 to 0.3 for CIFAR-100) are employed for regularization purposes on CIFAR-10/100. As for the evaluation on ImageNet, we train the 14-layer architecture from scratch for 250 epochs with a batch size of 1024. The learning rate is warmed up to 0.7 for the first 5 epochs and then decreased to zero with a cosine schedule. We adopt the SGD optimizer with 0.9 momentum and a weight decay of 10^{-5}.

The results on CIFAR-10/100 and ImageNet are summarized in Table 8 and Table 9, respectively. As shown in Table 8, both our HNAS (C10) and HNAS (C100) are capable of achieving state-of-the-art performance on CIFAR-10 and CIFAR-100, correspondingly, while incurring lower search costs than other training-based NAS algorithms. Even compared with other training-free NAS baselines, e.g., TE-NAS, our HNAS can still enjoy a compelling search cost. Overall, these results further validate that our HNAS is indeed able to enjoy the superior search efficiency of training-free NAS and also the remarkable search effectiveness of training-based NAS. More interestingly, our HNAS (C10) can achieve a lower test error on CIFAR-10 but a higher test error on CIFAR-100 when compared with HNAS (C100). This result indicates that similar to training-based NAS algorithms, directly searching on the target dataset is also able to improve the final performance in HNAS. By exploiting this advantage over other training-free NAS baselines, our HNAS thus is capable of selecting architectures achieving higher performances, as shown in Table 8. Similar results are also achieved on ImageNet as shown in Table 9. Overall, these results have further supported the superior search efficiency and remarkable search effectiveness of our HNAS that we have verified in Sec. 6.4.

Table 9: Performance comparison among SOTA image classi ers on ImageNet.

Algorithm	Test Error (%)		Params (M)	+ (M)	Search Cost (GPU Days)
	Top-1	Top-5			
Inception-v1 [64]	30.1	10.1	6.6	1448	-
MobileNet [65]	29.4	10.5	4.2	569	-
Shuf eNet 2 (v2) [66]	25.1	7.6	7.4	591	-
NASNet-A [25]	26.0	8.4	5.3	564	2000
AmoebaNet-A [26]	25.5	8.0	5.1	555	3150
PNAS [57]	25.8	8.1	5.1	588	225
MnasNet-92 [67]	25.2	8.0	4.4	388	-
DARTS [5]	26.7	8.7	4.7	574	4.0
SNAS (mild) [27]	27.3	9.2	4.3	522	1.5
GDAS [29]	26.0	8.5	5.3	581	0.21
ProxylessNAS [68]	24.9	7.5	7.1	465	8.3
DARTS- [61]	23.8	7.0	4.5	467	4.5
SDARTS-ADV [62]	25.2	7.8	5.4	594	1.3
DrNAS [30]	23.7	7.1	5.7	-	4.6
TE-NAS (C10) [7]	26.2	8.3	5.0	-	0.05
TE-NAS (ImageNet) [7]	24.5	7.5	5.4	-	0.17
NASI-ADA [8]	25.0	7.8	4.9	559	0.01
HNAS (C100)	24.8	7.8	5.2	601	0.1
HNAS (ImageNet)	24.3	7.4	5.1	575	0.5

C.6 Ablation Studies

Ablation Study on Initialization Method. While our theoretical analyses throughout this paper are based on the initialization using the standard normal distribution (Sec. 3.2), we wonder[4] whether our theoretical results are also applicable to DNNs using different initialization methods, e.g., Xavier [70] and Kaiming [71] initialization. Speci cally, we compare the correlation between the true generalization performances of all candidate architectures in NAS-Bench-201 and the generalization guarantees in Sec. 4.3 that are evaluated using different initialization methods. Table 10 summarizes the comparison. Here, we use the same parameters applied in Sec. 6.2 for Corollary 2. Notably, Table 10 shows that our generalization guarantees for training-free NAS, i.e., Corollary 1, 2, can also perform well for training-free NAS using DNNs initialized with different methods, indicating a wider application of our generalization guarantees in Sec. 4.3. Of note, LeCun initialization can achieve the best results among the three initialization methods in Table 10 since it satis es our assumption about the initialization of DNNs. As an implication, LeCun initialization is more preferred when using the training-free metrics from Sec. 3.2 to characterize the architecture performances in training-free NAS.

Ablation Study on Batch Size. Theoretically, the training-free metrics from Sec. 3.2 are de ned over the whole training dataset. In practice, we usually only apply a batch of randomly sampled data points to evaluate these training-free metrics in order to achieve a desirable computational ef ciency, which follows [9]. To investigate the impact of batch size on these metrics, we examine the correlation between the true generalization performances of all candidate architectures in NAS-Bench-201 and their generalization guarantees in the non-realizable scenario under varying batch sizes. Table 11 summarizes the results. Here, we use the same parameters applied in Sec. 6.2 for Corollary 2. Besides the impact of batch size on training-free metrics, we also include the impact of batch size on condition number in this table. Speci cally, in the upper part of Table 11, the correlations are evaluated using a batch size of 64 for and varying batch sizes for any training-free metric from Sec. 3.2. Meanwhile, in the lower part of Table 11, the correlations are evaluated using varying batch sizes for and a batch size of 64 for any training-free metric. Notably, Table 11 shows that similar results will be achieved even when training-free metrics are evaluated under varying batch sizes,

[4] Note that this initialization is equivalent to the LeCun initialization [69] according to [13].

Table 10: Correlation between the test errors (on CIFAR-10) of all architectures in NAS-Bench-201 and our generalization guarantees in Sec. 4.3 that are evaluated on DNNs using different initialization methods.

Initialization	Spearman				Kendall's Tau			
	M_{Grad}	M_{SNIP}	M_{GraSP}	M_{Trace}	M_{Grad}	M_{SNIP}	M_{GraSP}	M_{Trace}
Realizable scenario								
LeCun [69]	0.637	0.639	0.566	0.538	0.469	0.472	0.400	0.387
Xavier [70]	0.608	0.627	0.449	0.465	0.445	0.463	0.316	0.334
He [71]	0.609	0.615	0.340	0.460	0.446	0.454	0.242	0.334
Non-realizable scenario								
LeCun [69]	0.750	0.748	0.686	0.697	0.559	0.556	0.501	0.512
Xavier [70]	0.676	0.685	0.615	0.635	0.493	0.501	0.442	0.460
He [71]	0.607	0.611	0.505	0.569	0.436	0.439	0.358	0.407

Table 11: Correlation between the test errors (on CIFAR-10) of all architectures in NAS-Bench-201 and their generalization guarantees in the non-realizable scenario under varying batch size.

Batch Size	Spearman				Kendall's Tau			
	M_{Grad}	M_{SNIP}	M_{GraSP}	M_{Trace}	M_{Grad}	M_{SNIP}	M_{GraSP}	M_{Trace}
Batch size 64 for and varying batch sizes for any M								
4	0.737	0.741	0.671	0.684	0.547	0.550	0.487	0.501
8	0.739	0.743	0.676	0.689	0.549	0.552	0.492	0.506
16	0.747	0.748	0.685	0.690	0.556	0.556	0.499	0.507
32	0.750	0.748	0.687	0.690	0.558	0.556	0.502	0.506
64	0.750	0.748	0.686	0.697	0.559	0.556	0.501	0.512
Varying batch sizes for and batch size 64 for any M								
4	0.578	0.585	0.569	0.509	0.416	0.421	0.402	0.362
8	0.597	0.603	0.591	0.542	0.429	0.433	0.419	0.386
16	0.628	0.633	0.620	0.582	0.462	0.455	0.442	0.414
32	0.663	0.666	0.645	0.621	0.479	0.481	0.462	0.445
64	0.750	0.748	0.686	0.697	0.559	0.556	0.501	0.512

whereas evaluated under varying batch sizes will lead to different results, indicating that more sensitive to batch size than training-free metrics. As an implication, while a small batch size is also able to perform well in practice, a large batch size is more preferred when using our generalization guarantees for training-free NAS.

Ablation Study on Layer Width. While our theoretical analyses are based on over-parameterized DNNs, i.e., $n > N$ in our Theorem 2, we are also curious about how the layer width will influence our empirical results. In particular, we examine the correlation between the true generalization performances of all candidate architectures in NAS-Bench-201 and their generalization guarantee in the non-realizable scenario under varying layer width. Similar to the ablation study on batch size, we investigate the impacts of layer width on the training-free metrics from Sec. 3.2 and the condition number separately. Table 12 summarizes the results. Here, we use the same parameters applied in Sec. 6.2 for Corollary 2. As shown in Table 12, our generalization guarantee in the non-realizable scenario also performs well when layer width becomes smaller. Surprisingly, similar results can be achieved for training-free metrics evaluated under varying layer widths, whereas a larger layer width for training-free metrics typically leads to marginally higher correlations in Table 12. On the contrary, a larger layer width for leads to lower correlations in Table 12. This may result from the similar behavior that can be achieved by layer width and topology width since both layer width and topology width are used to measure the width of DNN but in totally different perspectives. Therefore, increasing layer width will make deep architectures (in terms of topology) more indistinguishable from wide architectures (in terms of topology) and hence make it harder to apply our generalization guarantee in Corollary 2 to characterize the architecture performances in a search space. As an

Table 12: Correlation between the test errors (on CIFAR-10) of all architectures in NAS-Bench-201 and their generalization guarantees in the non-realizable scenario under varying layer widths, which are measured by the number of initial channels in our experiments. Larger initial channels indicates a large layer width.

Init Channels	Spearman				Kendall's Tau			
	M_{Grad}	M_{SNIP}	M_{GraSP}	M_{Trace}	M_{Grad}	M_{SNIP}	M_{GraSP}	M_{Trace}
	4 channels for and varying channels for any M							
4	0.744	0.746	0.688	0.732	0.550	0.552	0.499	0.539
8	0.750	0.753	0.707	0.744	0.556	0.559	0.515	0.550
16	0.753	0.753	0.728	0.750	0.558	0.559	0.535	0.556
32	0.755	0.756	0.736	0.752	0.560	0.562	0.543	0.558
	Varying channels for and 32 channels for any M							
4	0.755	0.756	0.736	0.752	0.560	0.562	0.543	0.558
8	0.720	0.722	0.700	0.709	0.529	0.531	0.512	0.522
16	0.698	0.700	0.677	0.681	0.511	0.514	0.492	0.498
32	0.686	0.688	0.664	0.664	0.501	0.503	0.481	0.484

Table 13: Correlation between the test errors of all architectures in NAS-Bench-201 and our generalization guarantees in Sec. 4.3 using training-free metrics M_{KNAS}, M_{Fisher}, $M_{SynFlow}$ and M_{NASWOT} that are evaluated on various datasets. Each correlation is reported with the mean and standard deviation using the metrics evaluated on CIFAR-10/100 and ImageNet-16-120.

Dataset	Spearman				Kendall's Tau			
	M_{KNAS}	M_{Fisher}	$M_{SynFlow}$	M_{NASWOT}	M_{KNAS}	M_{Fisher}	$M_{SynFlow}$	M_{NASWOT}
	Realizable scenario							
C10	0.53 0.02	0.39 0.01	0.78 0.00	0.09 0.02	0.39 0.02	0.29 0.01	0.58 0.00	0.10 0.00
C100	0.53 0.03	0.39 0.01	0.76 0.00	0.09 0.02	0.38 0.02	0.29 0.01	0.57 0.00	0.11 0.01
IN-16	0.46 0.02	0.32 0.01	0.75 0.00	0.16 0.02	0.33 0.02	0.24 0.01	0.56 0.00	0.15 0.02
	Non-realizable scenario							
C10	0.66 0.02	0.51 0.00	0.81 0.00	0.05 0.00	0.49 0.02	0.37 0.00	0.61 0.00	0.03 0.00
C100	0.67 0.03	0.51 0.01	0.80 0.02	0.05 0.01	0.49 0.02	0.37 0.00	0.60 0.00	0.03 0.00
IN-16	0.62 0.04	0.44 0.00	0.78 0.00	0.05 0.01	0.45 0.03	0.32 0.00	0.59 0.00	0.03 0.00

implication, a large layer width for training-free metrics and a smaller layer width for condition number are more preferred when applying our generalization guarantees for training-free NAS in practice.

Ablation Study on Generalization Guarantees and HNAS Using Non-Gradient-Based Training-Free Metrics. As Appendix C.1 has validated that our Theorem 1 may also provide valid theoretical connections for certain non-gradient-based training-free metrics, we wonder whether our theoretical generalization guarantees and HNAS based on Theorem 1 are also applicable to these non-gradient-based training-free metrics. In particular, we first examine the correlation between the true generalization performances of all candidate architectures in NAS-Bench-201 and their generalization (Sec. 4.3) using training-free metrics M_{Fisher}, $M_{SynFlow}$ and M_{NASWOT}. Table 13 summarizes the results. Here, we use the same parameters applied in Sec. 6.2 for Corollary 2. While M_{Fisher} and $M_{SynFlow}$ enjoy higher correlation to M_{Trace} than M_{NASWOT} in Appendix C.1, our generalization guarantees also performs better when using M_{Fisher} and $M_{SynFlow}$. We then apply our HNAS based on these training-free metrics in NAS-Bench-201 and the Table 14 summarizes the search results. Similarly, our HNAS based on M_{Fisher} and $M_{SynFlow}$ can also find better-performing architectures than HNAS (M_{NASWOT}). Surprisingly, HNAS ($M_{SynFlow}$) can even achieve competitive results when compared with HNAS using gradient-based training-free metrics. These results therefore indicate that our HNAS sometimes may also be able to improve over training-free NAS using non-gradient-based training-free metrics especially when these non-gradient-based training-free metrics contain certain gradient information.

Table 14: Comparison among HNAS using different training-free metrics in NAS-Bench-201. The performance of each HNAS variant is reported with the mean and standard deviation of five independent searches and the search costs are evaluated on a single Nvidia 1080Ti.

Algorithm	**Test Accuracy (%)**						**Search Cost** (GPU Sec.)
	C10		C100		IN-16		
HNAS (M_{Grad})	94.04	0.21	71.75	1.04	45.91	0.88	3010
HNAS (M_{SNIP})	93.94	0.02	71.49	0.11	46.07	0.14	2976
HNAS (M_{GraSP})	94.13	0.13	72.59	0.82	46.24	0.38	3148
HNAS (M_{Trace})	94.07	0.10	72.30	0.70	45.93	0.37	3006
HNAS (M_{KNAS})	94.19	0.06	72.94	0.52	46.31	0.38	3081
HNAS (M_{Fisher})	93.28	0.73	69.42	1.36	42.85	2.09	3309
HNAS (M_{SynFlow})	94.13	0.00	72.50	0.00	45.47	0.00	3615
HNAS (M_{NASWOT})	92.10	0.62	66.81	0.32	39.26	0.72	2832
Optimal	94.37		73.51		47.31		-

Table 15: Comparison between HNAS and its training-free variant in NAS-Bench-201.

Algorithm	**Test Accuracy (%)**					
	C10		C100		IN-16	
$= M_{\text{Trace}}$	93.50		69.78		43.73	
HNAS (M_{Trace})	94.10	0.16	72.48	0.95	46.30	0.17
Optimal	94.37		73.51		47.31	

Ablation Study on Optimization Process of HNAS. In this section, we examine the evolution of the correlation between the test errors of candidate architectures in the NAS search space and their generalization guarantees in the non-realizable scenario with the increasing BO iterations in our HNAS framework. Figure 5 illustrates the results in NAS-Bench-201 with CIFAR-10 dataset and training-free metric M_{Trace}. Note that in every BO iteration of Figure 5, the Spearman correlation we reported corresponds to the pair of hyperparameters and that achieves the best validation performance in the query history. As shown in Figure 5, our HNAS framework, interestingly, is indeed selecting better-performing architectures by selecting hyperparameters and that can achieve higher Spearman correlation in the search space. These results therefore further justify the advantages of introducing BO algorithms with training-based performances into training-free NAS for better characterization.

Ablation Study on Training-Free Variant of HNAS. According to (7) in our main paper, a completely training-free metric can be produced by simply specifying the values of and with prior knowledge. For example, by setting $= 0$, we can obtain the training-free metric $= M_{\text{Trace}}$. However, obtaining prior knowledge regarding the best choice of and for NAS is non-trivial. Therefore, tuning and would be a better alternative to achieve more competitive search results in practice. To demonstrate this, we compare the performance of the architecture selected from training-free metric $= M_{\text{Trace}}$ vs. our standard HNAS framework in Table 15. Notably, the results in Table 15 demonstrate that tuning and based on training-based performances can indeed lead to improved search results and therefore will be a better alternative than pre-defining and for a completely training-free NAS, which further justifies the essence of combining training-free and training-based methods (as one of our major contributions) in HNAS.

Ablation Study on BO algorithm in HNAS. To investigate the influence of different BO algorithms (i.e., different acquisition functions) on the optimization part of our HNAS, we compare the search results obtained from using different acquisition functions (i.e., expected improvement (EI) vs. upper confidence bound (UCB)) with HNAS(M_{Trace}) and HNAS(M_{Grad}) on NAS-Bench-201 in Table 16. Since the default hyperparameters for different acquisition functions in [50] have already been tuned for a variety of tasks, we directly make use of them in our experiments without any changes.

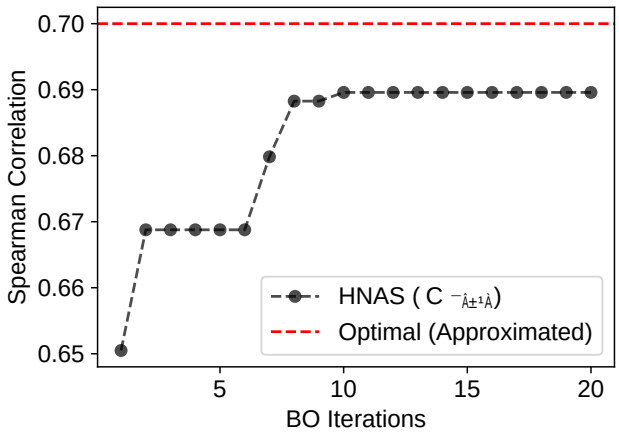

Figure 5: Evolution of the correlation between the test errors (on CIFAR-10) of all architectures in NAS-Bench-201 and their generalization guarantees (using M_{Trace}) in the non-realizable scenario with the BO iterations in our HNAS framework.

Table 16: Comparison among HNAS using different acquisition functions in NAS-Bench-201. The performance of each HNAS variant is reported with the mean and standard deviation of five independent searches.

Algorithm	**Test Accuracy (%)**					
	C10		C100		IN-16	
HNAS (M_{Grad}) w/ EI	94.04	0.21	71.75	1.04	45.91	0.88
HNAS (M_{Grad}) w/ UCB	94.05	0.18	72.04	1.18	45.81	0.88
HNAS (M_{Trace}) w/ EI	94.07	0.10	72.30	0.70	45.93	0.37
HNAS (M_{Trace}) w/ UCB	94.10	0.16	72.48	0.95	46.30	0.17
Optimal	94.37		73.51		47.31	

Interestingly, the results in Table 16 show that different acquisition functions (i.e., different BO algorithms) typically have limited influence on our HNAS framework. That is, our HNAS is shown to be relatively robust to the change of acquisition function in BO.