Appendix A Proofs

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

A.1 Proof of Theorem 1

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

$$
\begin{aligned}
\mathcal{M}_{\text{Grad}} &\overset{(a)}{=} \left\| \frac{1}{m} \sum_{i=1}^{m} \nabla_{\boldsymbol{\theta}} \ell(f(\boldsymbol{x}_i, \boldsymbol{\theta}_0), y_i) \right\|_2 \\
&\overset{(b)}{\leq} \frac{1}{m} \sum_{i=1}^{m} \left\| \nabla_{\boldsymbol{\theta}} \ell(f(\boldsymbol{x}_i, \boldsymbol{\theta}_0), y_i) \right\|_2 \\
&\overset{(c)}{\leq} \frac{1}{m} \sum_{i=1}^{m} \left| \nabla_f \ell(f(\boldsymbol{x}_i, \boldsymbol{\theta}_0), y_i) \right| \left\| \nabla_{\boldsymbol{\theta}} f(\boldsymbol{x}_i, \boldsymbol{\theta}_0) \right\|_2 \\
&\overset{(d)}{\leq} \frac{\beta}{m} \sum_{i=1}^{m} \left\| \nabla_{\boldsymbol{\theta}} f(\boldsymbol{x}_i, \boldsymbol{\theta}_0) \right\|_2 \\
&\overset{(e)}{\leq} \frac{\beta}{m} \sqrt{m \sum_{i=1}^{m} \left\| \nabla_{\boldsymbol{\theta}} f(\boldsymbol{x}_i, \boldsymbol{\theta}_0) \right\|_2^2} \\
&\overset{(f)}{=} \beta \mathcal{M}_{\text{Trace}}
\end{aligned}
\tag{8}
$$

where we let $\nabla_f \ell(f(\boldsymbol{x}_i, \boldsymbol{\theta}_0), y_i)$ be the gradient of the output of DNN model f. Note that (a) follows from the definition of $\mathcal{M}_{\text{Grad}}$ 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 $\mathcal{M}_{\text{Trace}}$ in Sec. 3.2, i.e.,

$$
\mathcal{M}_{\text{Trace}} = \sqrt{\frac{1}{m} \|\boldsymbol{\Theta}_0\|_{\text{tr}}} = \sqrt{\frac{1}{m} \sum_{i=1}^{m} \left\| \nabla_{\boldsymbol{\theta}} f(\boldsymbol{x}_i, \boldsymbol{\theta}_0) \right\|_2^2} \; . \tag{9}
$$

Let $C_1 \triangleq \beta$, we then have

$$
\mathcal{M}_{\text{Grad}} \leq C_1 \mathcal{M}_{\text{Trace}} \; . \tag{10}
$$

Connecting $\mathcal{M}_{\text{SNIP}}$ with $\mathcal{M}_{\text{Grad}}$. We firstly introduce the following lemma.

Lemma A.1 (Laurent and Massart [33]). *If* $\mathrm{x}_1, \cdots, \mathrm{x}_k$ *are independent standard normal random variables, for* $\mathrm{y} = \sum_{i=1}^{k} \mathrm{x}_i^2$ *and any* ϵ*,*

$$
\mathbb{P}(\mathrm{y} - k \geq 2\sqrt{k\epsilon} + 2\epsilon) \leq \exp(-\epsilon) \; .
$$

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

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

Using the results above and following the definition of $\mathcal{M}_{\text{Grad}}$, with probability at least $1-\delta$ over random initialization, we have

$$
\begin{aligned}
\mathcal{M}_{\text{SNIP}} &= \frac{1}{m}\sum_{i=1}^{m}\left|\boldsymbol{\theta}_0^\top \nabla_{\boldsymbol{\theta}}\mathcal{L}(f(\boldsymbol{x}_i,\boldsymbol{\theta}_0),y_i)\right| \\
&\le \frac{1}{m}\sum_{i=1}^{m}\|\boldsymbol{\theta}_0\|_2\|\nabla_{\boldsymbol{\theta}}\ell(f(\boldsymbol{x}_i,\boldsymbol{\theta}_0),y_i)\|_2 \\
&\le \sqrt{d+2\sqrt{d\ln\frac{1}{\delta}}+2\ln\frac{1}{\delta}}\cdot\frac{1}{m}\sum_{i=1}^{m}\|\nabla_{\boldsymbol{\theta}}\ell(f(\boldsymbol{x}_i,\boldsymbol{\theta}_0),y_i)\|_2 \\
&\le \beta\sqrt{d+2\sqrt{d\ln\frac{1}{\delta}}+2\ln\frac{1}{\delta}}\mathcal{M}_{\text{Trace}}\ .
\end{aligned}
\tag{12}
$$

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

$$
\mathcal{M}_{\text{SNIP}} \le C_2\mathcal{M}_{\text{Trace}}\ . \tag{13}
$$

Connecting $\mathcal{M}_{\text{GraSP}}$ and $\mathcal{M}_{\text{Grad}}$. We firstly introduce the following lemma adapted from [16].

Lemma A.2 (Lemma 1 in [16])**.** *Let $\delta\in(0,1)$. There exist the constant $\rho_1,\rho_2>0$ such that for any $r>0$, $\boldsymbol{\theta},\boldsymbol{\theta}'\in B(\boldsymbol{\theta}_0,r/\sqrt{n})$ and any input $\boldsymbol{x}$ within the dataset, with probability at least $1-\delta$ over random initialization, we have*

$$
\begin{gathered}
\|\nabla_{\boldsymbol{\theta}}f(\boldsymbol{x},\boldsymbol{\theta})\|_2 \le \rho_1 \\
\|\nabla_{\boldsymbol{\theta}}f(\boldsymbol{x},\boldsymbol{\theta})-\nabla_{\boldsymbol{\theta}'}f(\boldsymbol{x},\boldsymbol{\theta}')\|_2 \le \rho_2\|\boldsymbol{\theta}-\boldsymbol{\theta}'\|_2
\end{gathered}
$$

where $B(\boldsymbol{\theta}_0,r/\sqrt{n})\triangleq\{\boldsymbol{\theta}\mid\|\boldsymbol{\theta}-\boldsymbol{\theta}_0\|\le r/\sqrt{n}\}$.

To ease the notation, we use $\nabla_f\ell(f(\boldsymbol{x}_i,\boldsymbol{\theta}_0),y_i)$ to denote the gradient of the output (i.e., $f(\boldsymbol{x}_i,\boldsymbol{\theta}_0)$) from the DNN model f. According to the definition of Hessian matrix, $\mathbf{H}_i$ applied in $\mathcal{M}_{\text{GraSP}}$ can be computed as

$$
\begin{aligned}
\mathbf{H}_i &= \nabla^2_{\boldsymbol{\theta}_0}\ell(f(\boldsymbol{x}_i,\boldsymbol{\theta}_0),y_i) \\
&= \nabla_{\boldsymbol{\theta}}\left[\nabla_f\ell(f(\boldsymbol{x}_i,\boldsymbol{\theta}_0),y_i)\nabla_{\boldsymbol{\theta}}f(\boldsymbol{x}_i,\boldsymbol{\theta}_0)\right] \\
&= \nabla^2_f\ell(f(\boldsymbol{x}_i,\boldsymbol{\theta}_0),y_i)\nabla_{\boldsymbol{\theta}}f(\boldsymbol{x}_i,\boldsymbol{\theta}_0)\nabla_{\boldsymbol{\theta}}f(\boldsymbol{x}_i,\boldsymbol{\theta}_0)^\top+\nabla_f\ell(f(\boldsymbol{x}_i,\boldsymbol{\theta}_0),y_i)\nabla^2_{\boldsymbol{\theta}}f(\boldsymbol{x}_i,\boldsymbol{\theta}_0)\ .
\end{aligned}
\tag{14}
$$

Since $\ell(\cdot,\cdot)$ is assumed to be γ-Lipschitz smooth and β-Lipschitz continuous in the first argument, we can then bound the operator norm of this hessian matrix $\mathbf{H}_i$ induced by the input $\boldsymbol{x}_i$ in the dataset with

$$
\begin{aligned}
\|\mathbf{H}_i\|_2 &= \left\|\nabla^2_f\ell(f(\boldsymbol{x}_i,\boldsymbol{\theta}_0),y_i)\nabla_{\boldsymbol{\theta}}f(\boldsymbol{x}_i,\boldsymbol{\theta}_0)\nabla_{\boldsymbol{\theta}}f(\boldsymbol{x}_i,\boldsymbol{\theta}_0)^\top+\nabla_f\ell(f(\boldsymbol{x}_i,\boldsymbol{\theta}_0),y_i)\nabla^2_{\boldsymbol{\theta}}f(\boldsymbol{x}_i,\boldsymbol{\theta}_0)\right\|_2 \\
&\le \left|\nabla^2_f\ell(f(\boldsymbol{x}_i,\boldsymbol{\theta}_0),y_i)\right|\left\|\nabla_{\boldsymbol{\theta}}f(\boldsymbol{x}_i,\boldsymbol{\theta}_0)\nabla_{\boldsymbol{\theta}}f(\boldsymbol{x}_i,\boldsymbol{\theta}_0)^\top\right\|_2+\left|\nabla_f\ell(f(\boldsymbol{x}_i,\boldsymbol{\theta}_0),y_i)\right|\left\|\nabla^2_{\boldsymbol{\theta}}f(\boldsymbol{x}_i,\boldsymbol{\theta}_0)\right\|_2 \\
&\le \gamma\left\|\nabla_{\boldsymbol{\theta}}f(\boldsymbol{x}_i,\boldsymbol{\theta}_0)\nabla_{\boldsymbol{\theta}}f(\boldsymbol{x}_i,\boldsymbol{\theta}_0)^\top\right\|_2+\beta\left\|\nabla^2_{\boldsymbol{\theta}}f(\boldsymbol{x}_i,\boldsymbol{\theta}_0)\right\|_2 \\
&= \gamma\left\|\nabla_{\boldsymbol{\theta}}f(\boldsymbol{x}_i,\boldsymbol{\theta}_0)\right\|_2^2+\beta\left\|\nabla^2_{\boldsymbol{\theta}}f(\boldsymbol{x}_i,\boldsymbol{\theta}_0)\right\|_2 \\
&\le \gamma\rho_1^2+\beta\rho_2
\end{aligned}
\tag{15}
$$

where the last inequality results from Lemma A.2 and is satisfied with probability at least $1-\delta$ over random initialization.

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

$$\begin{aligned}\mathcal{M}_{\text{GraSP}} &= \frac{1}{m}\left|\sum_{i=1}^{m}\boldsymbol{\theta}_0^\top\left(\mathbf{H}_i\nabla_{\boldsymbol{\theta}}\mathcal{L}(f(\boldsymbol{x}_i,\boldsymbol{\theta}_0),y_i)\right)\right| \\ &\le \frac{1}{m}\|\boldsymbol{\theta}_0\|_2\sum_{i=1}^{m}\|\mathbf{H}_i\nabla_{\boldsymbol{\theta}}\mathcal{L}(f(\boldsymbol{x}_i,\boldsymbol{\theta}_0),y_i)\|_2 \\ &\le \frac{1}{m}\|\boldsymbol{\theta}_0\|_2\sum_{i=1}^{m}\|\mathbf{H}_i\|_2\|\nabla_{\boldsymbol{\theta}}\mathcal{L}(f(\boldsymbol{x}_i,\boldsymbol{\theta}_0),y_i)\|_2 \\ &\le (\gamma\rho_1^2+\beta\rho_2)\sqrt{d+2\sqrt{d\ln\frac{1}{\delta'}}+2\ln\frac{1}{\delta'}}\cdot\frac{1}{m}\sum_{i=1}^{m}\|\nabla_{\boldsymbol{\theta}}\ell(f(\boldsymbol{x}_i,\boldsymbol{\theta}_0),y_i)\|_2 \\ &\le \beta(\gamma\rho_1^2+\beta\rho_2)\sqrt{d+2\sqrt{d\ln\frac{1}{\delta'}}+2\ln\frac{1}{\delta'}}\mathcal{M}_{\text{Trace}}\ .\end{aligned} \tag{16}$$

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

$$\mathcal{M}_{\text{GraSP}} \le C_3\mathcal{M}_{\text{Trace}}\ , \tag{17}$$

which concludes our proof.

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

$$\mathcal{M}_{\text{KNAS}} \triangleq \sqrt{\left|\frac{1}{m^2}\sum_{i,j=1}^{m}\nabla_{\boldsymbol{\theta}}f(\boldsymbol{x}_i,\boldsymbol{\theta}_0)^\top\nabla_{\boldsymbol{\theta}}f(\boldsymbol{x}_j,\boldsymbol{\theta}_0)\right|}\ . \tag{18}$$

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 $\mathcal{M}_{\text{KNAS}}$ is also gradient-based. As a result, we can also theoretically connect $\mathcal{M}_{\text{KNAS}}$ with $\mathcal{M}_{\text{Trace}}$ in a similar way:

$$\begin{aligned}\mathcal{M}_{\text{KNAS}}^2 &= \left|\frac{1}{m^2}\sum_{i,j=1}^{m}\nabla_{\boldsymbol{\theta}}f(\boldsymbol{x}_i,\boldsymbol{\theta}_0)^\top\nabla_{\boldsymbol{\theta}}f(\boldsymbol{x}_j,\boldsymbol{\theta}_0)\right| \\ &\le \frac{1}{m^2}\sqrt{m^2\sum_{i,j=1}^{m}\left(\nabla_{\boldsymbol{\theta}}f(\boldsymbol{x}_i,\boldsymbol{\theta}_0)^\top\nabla_{\boldsymbol{\theta}}f(\boldsymbol{x}_j,\boldsymbol{\theta}_0)\right)^2} \\ &= \frac{1}{m}\left\|\boldsymbol{\Theta}_0\right\|_{\text{F}} \le \frac{1}{m}\left\|\boldsymbol{\Theta}_0\right\|_{\text{tr}} = \mathcal{M}_{\text{Trace}}^2\end{aligned} \tag{19}$$

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 $\|\cdot\|_{\text{F}} \le \|\cdot\|_{\text{tr}}$ while the last equality is obtained based on the definition of $\mathcal{M}_{\text{Trace}}$. Therefore, we have the following theoretical connection between $\mathcal{M}_{\text{KNAS}}$ and $\mathcal{M}_{\text{Trace}}$, which we will validate empirically in Appendix C.1.

$$\mathcal{M}_{\text{KNAS}} \le \mathcal{M}_{\text{Trace}}\ . \tag{20}$$

Consequently, the theoretical results and the HNAS framework in this paper are also applicable to the training-free metric $\mathcal{M}_{\text{KNAS}}$. 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 $\|\nabla_f^2\ell(f,\cdot)\|_2 \le c$ 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 $\mathcal{G}$ over dataset $S = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^m$ of size m is usually defined as

$$\mathcal{R}_S(\mathcal{G}) = \mathbb{E}_{\boldsymbol{\epsilon}\in\{\pm 1\}^m}\left[\sup_{g\in\mathcal{G}} \frac{1}{m}\sum_{i=1}^m \epsilon_i g(\boldsymbol{x}_i)\right] , \tag{21}$$

with $\epsilon_i \in \{\pm 1\}$. Let $\boldsymbol{\theta}_0$ be the initialized parameters of DNN model f, we then define the following hypotheses that will be used to prove our lemmas and theorems:

$$\mathcal{F} \triangleq \{\boldsymbol{x} \mapsto f(\boldsymbol{x}, \boldsymbol{\theta}_t) : t > 0\}, \quad \mathcal{F}^{\text{lin}} \triangleq \{\boldsymbol{x} \mapsto f(\boldsymbol{x}, \boldsymbol{\theta}_0) + \nabla_{\boldsymbol{\theta}} f(\boldsymbol{x}, \boldsymbol{\theta}_0)^\top (\boldsymbol{\theta}_t - \boldsymbol{\theta}_0) : t > 0\} \tag{22}$$

where $f_t \in \mathcal{F}$ and $f_t^{\text{lin}} \in \mathcal{F}^{\text{lin}}$ are the function determined by the DNN model f and its corresponding linearization at step t of their optimization, respectively. Of note, the $\boldsymbol{\theta}_t$ 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 [16])**.** *Let* $n_1 = \cdots = n_{L-1} = n$ *and assume* $\lambda_{\min}(\boldsymbol{\Theta}_\infty) > 0$. *There exist the constant* $c > 0$ *and* $N > 0$ *such that for any* $n > N$ *and any* $\boldsymbol{x} \in \mathbb{R}^{n_0}$ *with* $\|\boldsymbol{x}\|_2 \leq 1$, *the following holds with probability at least* $1 - \delta$ *over random initialization when applying gradient descent with learning rate* $\eta < \eta_0$,

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

Remark. According to [16], $\lambda_{\min}(\boldsymbol{\Theta}_\infty) > 0$ usually holds especially when any input $\boldsymbol{x}$ from dataset S satisfies $\|\boldsymbol{x}\|_2 = 1$. In practice, $\|\boldsymbol{x}\|_2 = 1$ can be achieved by normalizing each input $\boldsymbol{x}$ from real-world dataset using its norm $\|\boldsymbol{x}\|_2$, 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., $\mathcal{F}$) can also be bounded using its linearization model (i.e., $\mathcal{F}^{\text{lin}}$) 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*

$$\mathcal{R}_S(\mathcal{F}) \leq \mathcal{R}_S(\mathcal{F}^{\text{lin}}) + \frac{c}{\sqrt{n}} .$$

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

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

Following the definition of Rademacher complexity, we can bound the complexity of $\mathcal{F}$ by

$$\begin{aligned}
\mathcal{R}_S(\mathcal{F}) &= \mathbb{E}_{\boldsymbol{\epsilon}\in\{\pm 1\}^m}\left[\sup_{f\in\mathcal{F}} \frac{1}{m}\sum_{i=1}^m \epsilon_i f(\boldsymbol{x}_i, \boldsymbol{\theta})\right] \\
&\leq \mathbb{E}_{\boldsymbol{\epsilon}\in\{\pm 1\}^m}\left[\sup_{f^{\text{lin}}\in\mathcal{F}^{\text{lin}}} \frac{1}{m}\sum_{i=1}^m \left(\epsilon_i f^{\text{lin}}(\boldsymbol{x}_i) + \frac{c}{\sqrt{n}}\right)\right] \\
&\leq \mathbb{E}_{\boldsymbol{\epsilon}\in\{\pm 1\}^m}\left[\sup_{f^{\text{lin}}\in\mathcal{F}^{\text{lin}}} \frac{1}{m}\sum_{i=1}^m \epsilon_i f^{\text{lin}}(\boldsymbol{x}_i)\right] + \mathbb{E}_{\boldsymbol{\epsilon}\in\{\pm 1\}^m}\left[\frac{c}{\sqrt{n}}\right] \\
&\leq \mathcal{R}_S(\mathcal{F}^{\text{lin}}) + \frac{c}{\sqrt{n}} ,
\end{aligned} \tag{24}$$

which completes the proof. □

Lemma A.5. *Let $f(\mathbf{X},\boldsymbol{\theta}_0) \triangleq [f(\boldsymbol{x}_1,\boldsymbol{\theta}_0)\cdots f(\boldsymbol{x}_m,\boldsymbol{\theta}_0)]^\top$ and $\boldsymbol{y} \triangleq [y_1 \cdots y_m]^\top$ be the outputs of DNN model f at initialization and the target labels of a dataset $S = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^m$, respectively. Given MSE loss $\mathcal{L} = \sum_{i=1}^m \|f^{\text{lin}}(\boldsymbol{x}_i,\boldsymbol{\theta}) - y_i\|_2^2/(2m)$ and NTK matrix at initialization $\boldsymbol{\Theta}_0 = \nabla_{\boldsymbol{\theta}} f(\mathbf{X},\boldsymbol{\theta}_0)\nabla_{\boldsymbol{\theta}} f(\mathbf{X},\boldsymbol{\theta}_0)^\top$, assume $\lambda_{\min}(\boldsymbol{\Theta}_0) > 0$, for any $t > 0$, the following holds when applying gradient descent on $f^{\text{lin}}(\boldsymbol{x},\boldsymbol{\theta})$ with learning rate $\eta < m/\lambda_{\max}(\boldsymbol{\Theta}_0)$:*

$$\|\boldsymbol{\theta}_t - \boldsymbol{\theta}_0\|_2 \le \|\boldsymbol{\theta}_\infty - \boldsymbol{\theta}_0\|_2 = \sqrt{\widehat{\boldsymbol{y}}^\top \boldsymbol{\Theta}_0^{-1} \widehat{\boldsymbol{y}}}$$

where $\boldsymbol{\theta}_t$ denotes the parameters of f^{lin} at step t of its model training and $\widehat{\boldsymbol{y}} \triangleq \boldsymbol{y} - f(\mathbf{X},\boldsymbol{\theta}_0)$. Besides, $\lambda_{\max}(\boldsymbol{\Theta}_0)$ and $\lambda_{\min}(\boldsymbol{\Theta}_0)$ denote the maximum and minimum eigenvalue of matrix $\boldsymbol{\Theta}_0$.

Proof. Following the update of gradient descent on MSE with learning rate $\eta < m/\lambda_{\max}(\boldsymbol{\Theta}_0)$, we have

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \frac{\eta}{m}\nabla_{\boldsymbol{\theta}} f(\mathbf{X},\boldsymbol{\theta}_0)^\top \left(f^{\text{lin}}(\mathbf{X},\boldsymbol{\theta}_t) - \boldsymbol{y}\right) \ . \tag{25}$$

Note that $\nabla_{\boldsymbol{\theta}} f(\mathbf{X},\boldsymbol{\theta}_0)$ is a $m \times d$ matrix and $f(\mathbf{X},\boldsymbol{\theta}_0), f^{\text{lin}}(\mathbf{X},\boldsymbol{\theta}_0), \boldsymbol{y}$ are m-dimensional column vectors. By subtracting $\boldsymbol{\theta}_0$, multiplying $\nabla_{\boldsymbol{\theta}} f(\mathbf{X},\boldsymbol{\theta}_0)$ and then adding $f(\mathbf{X},\boldsymbol{\theta}_0)$ on both sides of the equality above, we achieve

$$f(\mathbf{X},\boldsymbol{\theta}_0) + \nabla_{\boldsymbol{\theta}} f(\mathbf{X},\boldsymbol{\theta}_0)(\boldsymbol{\theta}_{t+1} - \boldsymbol{\theta}_0) = f(\mathbf{X},\boldsymbol{\theta}_0) + \nabla_{\boldsymbol{\theta}} f(\mathbf{X},\boldsymbol{\theta}_0)(\boldsymbol{\theta}_t - \boldsymbol{\theta}_0) - \frac{\eta}{m}\boldsymbol{\Theta}_0\left(f^{\text{lin}}(\mathbf{X},\boldsymbol{\theta}_t) - \boldsymbol{y}\right) \ , \tag{26}$$

which can be simplified as

$$\begin{aligned} f^{\text{lin}}(\mathbf{X},\boldsymbol{\theta}_{t+1}) &= f^{\text{lin}}(\mathbf{X},\boldsymbol{\theta}_t) - \frac{\eta}{m}\boldsymbol{\Theta}_0\Big[f^{\text{lin}}(\mathbf{X},\boldsymbol{\theta}_t) - \boldsymbol{y}\Big] \\ &= \Big(\mathbf{I} - \frac{\eta}{m}\boldsymbol{\Theta}_0\Big) f^{\text{lin}}(\mathbf{X},\boldsymbol{\theta}_t) + \frac{\eta}{m}\boldsymbol{\Theta}_0\boldsymbol{y} \ . \end{aligned} \tag{27}$$

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

$$\begin{aligned} f^{\text{lin}}(\mathbf{X},\boldsymbol{\theta}_{t+1}) &\overset{(a)}{=} \Big(\mathbf{I} - \frac{\eta}{m}\boldsymbol{\Theta}_0\Big)^{t+1} f^{\text{lin}}(\mathbf{X},\boldsymbol{\theta}_0) + \sum_{j=0}^{t}\Big(\mathbf{I} - \frac{\eta}{m}\boldsymbol{\Theta}_0\Big)^j \Big(\frac{\eta}{m}\boldsymbol{\Theta}_0\boldsymbol{y}\Big) \\ &\overset{(b)}{=} \Big(\mathbf{I} - \frac{\eta}{m}\boldsymbol{\Theta}_0\Big)^{t+1} f(\mathbf{X},\boldsymbol{\theta}_0) + \Big[\mathbf{I} - (\mathbf{I} - \frac{\eta}{m}\boldsymbol{\Theta}_0)^{t+1}\Big]\Big(\frac{\eta}{m}\boldsymbol{\Theta}_0\Big)^{-1}\frac{\eta}{m}\boldsymbol{\Theta}_0\boldsymbol{y} \\ &\overset{(c)}{=} \Big(\mathbf{I} - \frac{\eta}{m}\boldsymbol{\Theta}_0\Big)^{t+1}\Big(f(\mathbf{X},\boldsymbol{\theta}_0) - \boldsymbol{y}\Big) + \boldsymbol{y} \end{aligned} \tag{28}$$

where (b) follows from the sum of geometric series for matrix with $\eta < m/\lambda_{\max}(\boldsymbol{\Theta}_0)$ as well as the fact that $f^{\text{lin}}(\mathbf{X},\boldsymbol{\theta}_0) = f(\mathbf{X},\boldsymbol{\theta}_0)$. Note that this result can be integrated into (25) and provide the following explicit form of $\boldsymbol{\theta}_{t+1} - \boldsymbol{\theta}_0$ after applying gradient descent for $t+1$ times:

$$\begin{aligned} \boldsymbol{\theta}_{t+1} - \boldsymbol{\theta}_0 &= \sum_{k=0}^{t}\boldsymbol{\theta}_{k+1} - \boldsymbol{\theta}_k \\ &= \frac{\eta}{m}\nabla_{\boldsymbol{\theta}} f(\mathbf{X},\boldsymbol{\theta}_0)^\top \sum_{k=0}^{t}\Big(\mathbf{I} - \frac{\eta}{m}\boldsymbol{\Theta}_0\Big)^k (\boldsymbol{y} - f(\mathbf{X},\boldsymbol{\theta}_0)) \\ &= \frac{\eta}{m}\nabla_{\boldsymbol{\theta}} f(\mathbf{X},\boldsymbol{\theta}_0)^\top \sum_{k=0}^{t}(\mathbf{I} - \frac{\eta}{m}\boldsymbol{\Theta}_0)^k \widehat{\boldsymbol{y}} \end{aligned} \tag{29}$$

Since $\boldsymbol{\Theta}_0$ is symmetric, we can alternatively represent $\boldsymbol{\Theta}_0$ as $\boldsymbol{\Theta}_0 = \mathbf{V}\boldsymbol{\Lambda}\mathbf{V}^\top$ using principal component analysis (PCA) where $\mathbf{V}$ and $\boldsymbol{\Lambda}$ denotes the matrix of eigenvectors $\{\boldsymbol{v}_i\}_{i=1}^m$ and eigenvalues

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

$$\begin{aligned}
\|\boldsymbol{\theta}_{t+1}-\boldsymbol{\theta}_0\|_2 &= \sqrt{(\boldsymbol{\theta}_{t+1}-\boldsymbol{\theta}_0)^\top(\boldsymbol{\theta}_{t+1}-\boldsymbol{\theta}_0)} \\
&= \frac{\eta}{m}\sqrt{\widehat{\boldsymbol{y}}^\top \sum_{k=0}^{t}(\mathbf{I}-\frac{\eta}{m}\boldsymbol{\Theta}_0)^k \nabla_{\boldsymbol{\theta}} f(\mathbf{X},\boldsymbol{\theta}_0)\nabla_{\boldsymbol{\theta}} f(\mathbf{X},\boldsymbol{\theta}_0)^\top \sum_{k'=0}^{t}(\mathbf{I}-\frac{\eta}{m}\boldsymbol{\Theta}_0)^{k'}\widehat{\boldsymbol{y}}} \\
&= \frac{\eta}{m}\sqrt{\widehat{\boldsymbol{y}}^\top \sum_{k=0}^{t}(\mathbf{I}-\frac{\eta}{m}\boldsymbol{\Theta}_0)^k \boldsymbol{\Theta}_0 \sum_{k'=0}^{t}(\mathbf{I}-\frac{\eta}{m}\boldsymbol{\Theta}_0)^{k'}\widehat{\boldsymbol{y}}} \\
&= \frac{\eta}{m}\sqrt{\widehat{\boldsymbol{y}}^\top \sum_{k=0}^{t}(\mathbf{I}-\frac{\eta}{m}\mathbf{V}\boldsymbol{\Lambda}\mathbf{V}^\top)^k \mathbf{V}\boldsymbol{\Lambda}\mathbf{V}^\top \sum_{k'=0}^{t}(\mathbf{I}-\frac{\eta}{m}\mathbf{V}\boldsymbol{\Lambda}\mathbf{V}^\top)^{k'}\widehat{\boldsymbol{y}}} \\
&= \frac{\eta}{m}\sqrt{\widehat{\boldsymbol{y}}^\top \mathbf{V}\sum_{k=0}^{t}(\mathbf{I}-\frac{\eta}{m}\boldsymbol{\Lambda})^k \mathbf{V}^\top\mathbf{V}\boldsymbol{\Lambda}\mathbf{V}^\top\mathbf{V} \sum_{k'=0}^{t}(\mathbf{I}-\frac{\eta}{m}\boldsymbol{\Lambda})^{k'}\mathbf{V}^\top\widehat{\boldsymbol{y}}} \\
&= \frac{\eta}{m}\sqrt{\widehat{\boldsymbol{y}}^\top \mathbf{V}\sum_{k=0}^{t}(\mathbf{I}-\frac{\eta}{m}\boldsymbol{\Lambda})^k \boldsymbol{\Lambda} \sum_{k'=0}^{t}(\mathbf{I}-\frac{\eta}{m}\boldsymbol{\Lambda})^{k'}\mathbf{V}^\top\widehat{\boldsymbol{y}}} \\
&= \frac{\eta}{m}\sqrt{\sum_{i=1}^{m}\lambda_i\left[\sum_{k=0}^{t}(1-\frac{\eta}{m}\lambda_i)^k\right]^2(\boldsymbol{v}_i^\top\widehat{\boldsymbol{y}})^2}\,.
\end{aligned} \tag{30}$$

Since $\eta < m/\lambda_{\max}(\boldsymbol{\Theta}_0)$ and $\lambda_{\min}(\boldsymbol{\Theta}_0) > 0$, we have $0 < 1-\eta\lambda_i/m < 1$ and hence

$$\begin{aligned}
\|\boldsymbol{\theta}_{t}-\boldsymbol{\theta}_0\|_2 &= \frac{\eta}{m}\sqrt{\sum_{i=1}^{m}\lambda_i\left[\sum_{k=0}^{t-1}(1-\frac{\eta}{m}\lambda_i)^k\right]^2(\boldsymbol{v}_i^\top\widehat{\boldsymbol{y}})^2} \\
&\le \frac{\eta}{m}\sqrt{\sum_{i=1}^{m}\lambda_i\left[\sum_{k=0}^{t}(1-\frac{\eta}{m}\lambda_i)^k\right]^2(\boldsymbol{v}_i^\top\widehat{\boldsymbol{y}})^2} \\
&= \|\boldsymbol{\theta}_{t+1}-\boldsymbol{\theta}_0\|_2
\end{aligned} \tag{31}$$

We complete the proof by recursively applying the inequalities above

$$\begin{aligned}
\|\boldsymbol{\theta}_{t}-\boldsymbol{\theta}_0\|_2 &\le \|\boldsymbol{\theta}_{\infty}-\boldsymbol{\theta}_0\|_2 \\
&= \frac{\eta}{m}\sqrt{\sum_{i=1}^{m}\lambda_i\left[\sum_{k=0}^{\infty}(1-\frac{\eta}{m}\lambda_i)^k\right]^2(\boldsymbol{v}_i^\top\widehat{\boldsymbol{y}})^2} \\
&= \frac{\eta}{m}\sqrt{\sum_{i=1}^{m}\lambda_i\left[\frac{1}{\eta\lambda_i/m}\right]^2(\boldsymbol{v}_i^\top\widehat{\boldsymbol{y}})^2} \\
&= \sqrt{\sum_{i=1}^{m}\lambda_i^{-1}(\boldsymbol{v}_i^\top\widehat{\boldsymbol{y}})^2} \\
&= \sqrt{\widehat{\boldsymbol{y}}^\top\boldsymbol{\Theta}_0^{-1}\widehat{\boldsymbol{y}}}
\end{aligned} \tag{32}$$

□

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

$S \triangleq (\boldsymbol{x}_1, \cdots, \boldsymbol{x}_m)$ *admits the following upper bounds:*

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

where $\mathbf{X}$ *is the* $d \times m$*-matrix with* $\boldsymbol{x}_i$*s as columns:* $\mathbf{X} \triangleq [\boldsymbol{x}_1 \cdots \boldsymbol{x}_m]$.

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., $\mathcal{F}$) using its linearization model (i.e., $\mathcal{F}^{\text{lin}}$). 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

$$\begin{aligned}
\mathcal{R}_S(\mathcal{F}) &\overset{(a)}{\leq} \mathcal{R}_S(\mathcal{F}^{\text{lin}}) + \frac{c}{\sqrt{n}} \\
&\overset{(b)}{=} \mathbb{E}_{\boldsymbol{\epsilon} \in \{\pm 1\}^m} \left[\sup_{t \geq 0} \frac{1}{m} \sum_{i=1}^m \epsilon_i \left(f(\boldsymbol{x}_i, \boldsymbol{\theta}_0) + \nabla_{\boldsymbol{\theta}} f(\boldsymbol{x}_i, \boldsymbol{\theta}_0)^\top (\boldsymbol{\theta}_t - \boldsymbol{\theta}_0) \right) \right] + \frac{c}{\sqrt{n}} \\
&\overset{(c)}{=} \mathbb{E}_{\boldsymbol{\epsilon} \in \{\pm 1\}^m} \left[\sup_{t \geq 0} \frac{1}{m} \sum_{i=1}^m \epsilon_i \nabla_{\boldsymbol{\theta}} f(\boldsymbol{x}_i, \boldsymbol{\theta}_0)^\top (\boldsymbol{\theta}_t - \boldsymbol{\theta}_0) \right] + \frac{1}{m} \sum_{i=1}^m \mathbb{E}_{\boldsymbol{\epsilon} \in \{\pm 1\}^m} [\epsilon_i] \, f(\boldsymbol{x}_i, \boldsymbol{\theta}_0) + \frac{c}{\sqrt{n}} \\
&\overset{(d)}{\leq} \frac{\|\boldsymbol{\theta}_\infty - \boldsymbol{\theta}_0\|_2 \|\nabla_{\boldsymbol{\theta}} f(\mathbf{X}, \boldsymbol{\theta}_0)\|_{2,2}}{m} + \frac{c}{\sqrt{n}} \\
&\overset{(e)}{\leq} \frac{\|\nabla_{\boldsymbol{\theta}} f(\mathbf{X}, \boldsymbol{\theta}_0)\|_{2,2}}{m} \sqrt{\widehat{\boldsymbol{y}}^\top \boldsymbol{\Theta}_0^{-1} \widehat{\boldsymbol{y}}} + \frac{c}{\sqrt{n}} \\
&\overset{(f)}{\leq} \sqrt{\kappa \lambda_0} \cdot \sqrt{\frac{\widehat{\boldsymbol{y}}^\top \boldsymbol{\Theta}_0^{-1} \widehat{\boldsymbol{y}}}{m}} + \frac{c}{\sqrt{n}}
\end{aligned} \tag{33}$$

where (d) derives from Lemma A.6 and (f) derives from the following inequalities based on the definition $\kappa \triangleq \lambda_{\max}(\boldsymbol{\Theta}_0)/\lambda_{\min}(\boldsymbol{\Theta}_0)$ and $\lambda_0 \triangleq \lambda_{\min}(\boldsymbol{\Theta}_0)$.

$$\begin{aligned}
\|\nabla_{\boldsymbol{\theta}} f(\mathbf{X}, \boldsymbol{\theta}_0)\|_{2,2} &= \sqrt{\sum_{i=1}^m \|\nabla_{\boldsymbol{\theta}} f(\boldsymbol{x}_i, \boldsymbol{\theta}_0)\|_2^2} \\
&= \sqrt{\sum_{i=1}^m \lambda_i(\boldsymbol{\Theta}_0)} \\
&\leq \sqrt{m \kappa \lambda_0} \, .
\end{aligned} \tag{34}$$

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

Define the generalization error on the data distribution $\mathcal{D}$ as $\mathcal{L}_{\mathcal{D}}(g) \triangleq \mathbb{E}_{(\boldsymbol{x},y) \sim \mathcal{D}} \ell(g(\boldsymbol{x}), y)$ and the empirical error on the dataset $S = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^m$ that is randomly sampled from $\mathcal{D}$ as $\mathcal{L}_S(g) \triangleq \sum_{i=1}^m \ell(g(\boldsymbol{x}_i), y_i)$. Given the loss function $\ell(\cdot, \cdot)$ and the Rademacher complexity of any hypothesis class $\mathcal{G}$, the generalization error on the hypothesis class $\mathcal{G}$ 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* β*-Lipschitz continuous in the first argument. Then with probability at least* $1 - \delta$ *over dataset* S *of size* m*:*

$$\sup_{g \in \mathcal{G}} \{\mathcal{L}_{\mathcal{D}}(g) - \mathcal{L}_S(g)\} \leq 2\beta \mathcal{R}_S(\mathcal{G}) + 3\sqrt{\ln(2/\delta)/(2m)} \, .$$

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

$$\|\mathbf{A}\|_{\text{tr}} \left\|\mathbf{A}^{-1}\right\|_{\text{tr}} \leq m^2 \kappa \, .$$

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

$$\frac{\lambda_m}{\kappa} \le \lambda_i \le \lambda_1 \kappa \,. \tag{35}$$

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

$$\|\mathbf{A}\|_{\text{tr}} \left\|\mathbf{A}^{-1}\right\|_{\text{tr}} = (\sum_{i=1}^m \lambda_i) \cdot (\sum_{i=1}^m \lambda_i^{-1}) \le (m\lambda_1\kappa) \cdot \frac{m\kappa}{\lambda_m} = \frac{m^2\kappa^2}{\kappa} = m^2\kappa \,, \tag{36}$$

which concludes the proof. □

We are now able to prove Theorem 2 by combining the results in Lemma A.7 and (33). Specifically, under the conditions in Theorem A.3 and Lemma A.5, there exist constant $c, N > 0$ such that for any $f_t \in \mathcal{F}$ and any $n > N$, the following holds with probability at least $1 - 2\delta$ over random initialization,

$$\begin{aligned} \mathcal{L}_{\mathcal{D}}(f_t) &\le \mathcal{L}_S(f_t) + 2\beta\mathcal{R}_S(\mathcal{F}) + 3\sqrt{\frac{\ln(2/\delta)}{2m}} \\ &\le \mathcal{L}_S(f_t) + 2\beta\sqrt{\kappa\lambda_0} \cdot \sqrt{\frac{\widehat{\boldsymbol{y}}^\top \mathbf{\Theta}_0^{-1} \widehat{\boldsymbol{y}}}{m}} + \frac{2\beta c}{\sqrt{n}} + 3\sqrt{\frac{\ln(2/\delta)}{2m}} \,. \end{aligned} \tag{37}$$

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

$$\widehat{\boldsymbol{y}}^\top \mathbf{\Theta}_0^{-1} \widehat{\boldsymbol{y}} = \sum_{i=1}^m \frac{(\boldsymbol{v}_i^\top \widehat{\boldsymbol{y}})^2}{\lambda_i} \le \sum_{i=1}^m \frac{\|\boldsymbol{v}_i\|_2^2 \|\widehat{\boldsymbol{y}}\|_2^2}{\lambda_i} \le \sum_{i=1}^m \frac{m}{\lambda_i} \,. \tag{38}$$

Based on the fact that $\|\mathbf{\Theta}_0\|_{\text{tr}} = \sum_{i=1}^m \lambda_i$ and Lemma A.8, we finally achieve

$$\sqrt{\frac{\widehat{\boldsymbol{y}}^\top \mathbf{\Theta}_0^{-1} \widehat{\boldsymbol{y}}}{m}} \le \sqrt{\left\|\mathbf{\Theta}_0^{-1}\right\|_{\text{tr}}} \le \frac{m\sqrt{\kappa}}{\sqrt{\|\mathbf{\Theta}_0\|_{\text{tr}}}} = \frac{\sqrt{m\kappa}}{\mathcal{M}_{\text{Trace}}} \,. \tag{39}$$

By introducing (39) into (37), with $\lambda_0 \le 1$, we have

$$\mathcal{L}_{\mathcal{D}}(f_t) \le \mathcal{L}_S(f_t) + \frac{2\beta\kappa\sqrt{m}}{\mathcal{M}_{\text{Trace}}} + \frac{2\beta c}{\sqrt{n}} + 3\sqrt{\frac{\ln(2/\delta)}{2m}} \,. \tag{40}$$

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

$$\mathcal{L}_{\mathcal{D}}(f_t) \le \mathcal{L}_S(f_t) + \mathcal{O}(\kappa/\mathcal{M}) \,, \tag{41}$$

which finally concludes our proof of Theorem 2.

Remark. Our (41) still holds when $\lambda_0 \le z (z \neq 1)$, 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 firstly consider the convergence of f_t^{lin} under the same conditions in Theorem 2. Specifically, following the notations and results in Lemma A.5, let $\{\boldsymbol{v}_i\}_{i=1}^m$ and $\{\lambda_i\}_{i=1}^m$

be the eigenvectors and eigenvalues of $\mathbf{\Theta}_0$, respectively, we have

$$\begin{aligned}
\mathcal{L}_S(f_t^{\text{lin}}) &\overset{(a)}{=} \frac{1}{2m} \left\| f^{\text{lin}}(\mathbf{X}, \boldsymbol{\theta}_t) - \boldsymbol{y} \right\|_2^2 \\
&\overset{(b)}{=} \frac{1}{2m} \left\| \left(\mathbf{I} - \frac{\eta}{m}\mathbf{\Theta}_0\right)^t (f(\mathbf{X}, \boldsymbol{\theta}_0) - \boldsymbol{y}) \right\|_2^2 \\
&\overset{(c)}{=} \frac{1}{2m} \left\| \left(\mathbf{I} - \frac{\eta}{m}\mathbf{\Theta}_0\right)^t \widehat{\boldsymbol{y}} \right\|_2^2 \\
&\overset{(d)}{=} \frac{1}{2m} \sum_{i=1}^{m} \left(1 - \frac{\eta}{m}\lambda_i\right)^{2t} \left(\boldsymbol{v}_i^\top \widehat{\boldsymbol{y}}\right)^2 \\
&\overset{(e)}{\leq} \frac{1}{2m} \sum_{i=1}^{m} \left(1 - \frac{\eta}{m}\lambda_i\right)^{2t} \|\boldsymbol{v}_i\|_2^2 \, \|\widehat{\boldsymbol{y}}\|_2^2
\end{aligned} \tag{42}$$

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

$$\begin{aligned}
\mathcal{L}_S(f_t^{\text{lin}}) &\overset{(a)}{\leq} \frac{1}{2} \sum_{i=1}^{m} \left(1 - \frac{\eta}{m}\lambda_i\right)^{2t} \\
&\overset{(b)}{\leq} \frac{1}{2} \left(\sum_{i=1}^{m} 1 - \frac{\eta}{m}\lambda_i\right)^{2t} \\
&\overset{(c)}{=} \frac{1}{2} \left(m - \frac{\eta}{m}\|\mathbf{\Theta}_0\|_{\text{tr}}\right)^{2t} \\
&\overset{(d)}{=} \frac{1}{2} \left(m - \eta\mathcal{M}_{\text{Trace}}^2\right)^{2t} \\
&\overset{(e)}{\leq} \frac{1}{2} \left(m - \eta\mathcal{M}^2/C\right)^{2t}
\end{aligned} \tag{43}$$

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

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

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

$$\left|\mathcal{L}_S(f_t) - \mathcal{L}_S(f_t^{\text{lin}})\right| \leq \left|f_t - f_t^{\text{lin}}\right| \leq \mathcal{O}(\frac{1}{\sqrt{n}}) \, . \tag{45}$$

By introducing the results above into our Theorem 2 with $1/\sqrt{n}$ being absorbed in $\mathcal{O}(\cdot)$, we finally achieve the following results with a high probability,

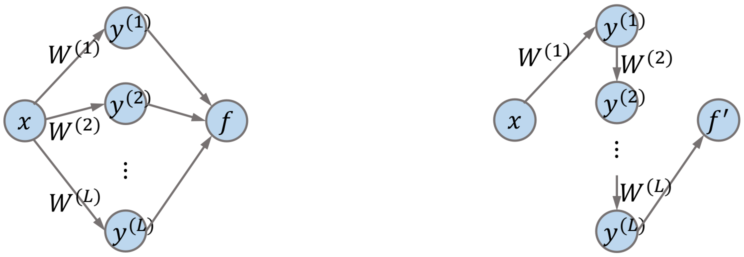
$$\begin{aligned}
\mathcal{L}_{\mathcal{D}}(f_t) &\leq \mathcal{L}_S(f_t) + \mathcal{O}(\kappa/\mathcal{M}) \leq \mathcal{L}_S(f_t^{\text{lin}}) + \mathcal{O}(\kappa/\mathcal{M}) \\
&\leq \frac{1}{2}\left(m - \eta\mathcal{M}^2/C\right)^{2t} + \mathcal{O}(\kappa/\mathcal{M}) \, ,
\end{aligned} \tag{46}$$

which thus concludes our proof.

A.4 Proof of Theorem 3

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

$$\begin{aligned}
\nabla_{\mathbf{W}_{j\cdot}^{(i)}} f(\boldsymbol{x}) &= \boldsymbol{x} \\
\nabla_{\mathbf{W}_{j\cdot}^{(i)}} f'(\boldsymbol{x}) &= \left(\prod_{k'=1}^{i-1} \mathbf{W}^{(k')}\boldsymbol{x}\right) \mathbf{1}^\top \left(\prod_{k=i+1}^{L} \mathbf{W}^{(k)}\right)_{\cdot j}
\end{aligned} \tag{47}$$



(a) Wide architecture

(b) Deep architecture

Figure 3: Two different architecture topologies for our analysis.

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

$$\left(\prod_{k=i+1}^{L}\mathbf{W}^{(k)}\right)_{\cdot j} \triangleq \left(\mathbf{W}^{(i+1)}\cdots\mathbf{W}^{(L)}\right)_{\cdot j} = \mathbf{W}^{(L)}\mathbf{W}^{(L-1)}\cdots\mathbf{W}^{(i+1)}_{\cdot j}\ , \tag{48}$$

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

$$\begin{aligned}\boldsymbol{\Theta}_0(\boldsymbol{x},\boldsymbol{x}') &= \sum_{i=1}^{L}\sum_{j=1}^{n}\left(\nabla_{\mathbf{W}^{(i)}_{j\cdot}}f(\boldsymbol{x})\right)^\top\nabla_{\mathbf{W}^{(i)}_{j\cdot}}f(\boldsymbol{x}') \\ &= \sum_{i=1}^{L}\sum_{j=1}^{n}\boldsymbol{x}^\top\boldsymbol{x}' = nL\cdot\boldsymbol{x}^\top\boldsymbol{x}'\ .\end{aligned} \tag{49}$$

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

$$\begin{aligned}\boldsymbol{\Theta}_0'(\boldsymbol{x},\boldsymbol{x}') &= \sum_{i=1}^{L}\sum_{j=1}^{n}\left(\nabla_{\mathbf{W}^{(i)}_{j\cdot}}f'(\boldsymbol{x})\right)^\top\nabla_{\mathbf{W}^{(i)}_{j\cdot}}f'(\boldsymbol{x}) \\ &= \sum_{i=1}^{L}\sum_{j=1}^{n}\left(\left(\prod_{k'=1}^{i-1}\mathbf{W}^{(k')}\boldsymbol{x}\right)\mathbf{1}^\top\left(\prod_{k=i+1}^{L}\mathbf{W}^{(k)}\right)_{\cdot j}\right)^\top\left(\prod_{k'=1}^{i-1}\mathbf{W}^{(k')}\boldsymbol{x}'\right)\mathbf{1}^\top\left(\prod_{k=i+1}^{L}\mathbf{W}^{(k)}\right)_{\cdot j} \\ &= \sum_{i=1}^{L}\sum_{j=1}^{n}\left(\mathbf{1}^\top\left(\prod_{k=i+1}^{L}\mathbf{W}^{(k)}\right)_{\cdot j}\right)^2\boldsymbol{x}^\top\left(\prod_{k'=1}^{i-1}\mathbf{W}^{(k')}\right)^\top\left(\prod_{k'=1}^{i-1}\mathbf{W}^{(k')}\right)\boldsymbol{x}' \\ &= \boldsymbol{x}^\top\sum_{i=1}^{L}\sum_{j=1}^{n}\left(\mathbf{1}^\top\left(\prod_{k=i+1}^{L}\mathbf{W}^{(k)}\right)_{\cdot j}\right)^2\left(\prod_{k'=1}^{i-1}\mathbf{W}^{(k')}\right)^\top\left(\prod_{k'=1}^{i-1}\mathbf{W}^{(k')}\right)\boldsymbol{x}'\ .\end{aligned} \tag{50}$$

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

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

Similarly, we also have

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

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

$$\begin{aligned}
\mathbb{E}\left[\boldsymbol{\Theta}_0'(\boldsymbol{x},\boldsymbol{x}')\right] &= \boldsymbol{x}^\top\mathbb{E}\left[\sum_{i=1}^{L}\sum_{j=1}^{n}\left(\mathbf{1}^\top\left(\prod_{k=i+1}^{L}\mathbf{W}^{(k)}\right)_{\cdot j}\right)^2\left(\prod_{k'=1}^{i-1}\mathbf{W}^{(k')}\right)^\top\left(\prod_{k'=1}^{i-1}\mathbf{W}^{(k')}\right)\right]\boldsymbol{x}' \\
&= \boldsymbol{x}^\top\left(\sum_{i=1}^{L}\sum_{j=1}^{n}\mathbb{E}\left[\left(\mathbf{1}^\top\left(\prod_{k=i+1}^{L}\mathbf{W}^{(k)}\right)_{\cdot j}\right)^2\right]\mathbb{E}\left[\left(\prod_{k'=1}^{i-1}\mathbf{W}^{(k')}\right)^\top\prod_{k'=1}^{i-1}\mathbf{W}^{(k')}\right]\right)\boldsymbol{x}' \\
&= \boldsymbol{x}^\top\left(\sum_{i=1}^{L}\sum_{j=1}^{n}n^{L-i}\cdot n^{i-1}\mathbf{I}\right)\boldsymbol{x}' \\
&= Ln^L\boldsymbol{x}^\top\boldsymbol{x}'\ .
\end{aligned} \tag{53}$$

By exploiting the fact that $\mathbf{X}^\top\mathbf{X} = \mathbf{I}$ with $\mathbf{X} \triangleq [\boldsymbol{x}_1\boldsymbol{x}_2\cdots\boldsymbol{x}_m]$, we finally conclude the proof by

$$\begin{aligned}
\boldsymbol{\Theta}_0(\mathbf{X},\mathbf{X}) &= Ln\cdot\mathbf{I} \\
\mathbb{E}\left[\boldsymbol{\Theta}_0'(\mathbf{X},\mathbf{X})\right] &= Ln^L\cdot\mathbf{I}\ .
\end{aligned} \tag{54}$$

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 μ and ν) from all sampled architectures. Meanwhile, following the common practice in [9], 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 μ and ν) in our Algorithm 1, BO will naturally be a better choice to find the optimal μ and ν 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 $u(\mu, \nu)$ and variance $\sigma^2(\mu, \nu)$ 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 $(\mu, \nu)^\top$ and the function value y in [48] be $\mathcal{L}_{\text{val}}(\mathcal{A})$) using the historical evaluations $\mathcal{H}_{k-1} = \{((\mu_i, \nu_i), \mathcal{L}_{\text{val}}(A_i^*))\}_{i=1}^{k-1}$ (this corresponds to line 6 in Algorithm 1 for iteration $k-1$). [2] Then, the mean $u(\mu, \nu)$ and standard deviation $\sigma(\mu, \nu)$ 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) $u(\mu, \nu) + \sqrt{\beta}\sigma(\mu, \nu)$ from [48] where the parameter $\beta > 0$ is set to trade off between exploitation vs. exploration for guaranteeing no regret asymptotically with high probability. Finally, an input (i.e., μ_k, ν_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., $(\mu_k, \nu_k) = \arg\max_{(\mu,\nu)} u(\mu, \nu) + \sqrt{\beta}\sigma(\mu, \nu)$ 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., $2000 \times k$ 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, and *(b)* 20 architectures with evaluated validation performance which can already allow our HNAS to select architectures achieving competitive performances. Overall, our (7) and 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 $\mathcal{M}_{\text{Trace}}$ 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 $\mathcal{M}_{\text{GraSP}}$ 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 $\mathcal{M}_{\text{GraSP}}$. To figure out *whether our Theorem 1 is also applicable to non-gradient-based training-free metrics*, we then provide the correlation between $\mathcal{M}_{\text{Fisher}}$ [51], $\mathcal{M}_{\text{SynFlow}}$ [52], $\mathcal{M}_{\text{NASWOT}}$ [6] and $\mathcal{M}_{\text{Trace}}$ [8] for the comparison. Interestingly, both $\mathcal{M}_{\text{Fisher}}$ and $\mathcal{M}_{\text{SynFlow}}$ achieve higher positive correlations with $\mathcal{M}_{\text{Trace}}$ than $\mathcal{M}_{\text{NASWOT}}$ in general. According to their mathematical forms in the corresponding papers, such a phenomenon may result from the fact that $\mathcal{M}_{\text{Fisher}}$ and $\mathcal{M}_{\text{SynFlow}}$ have contained certain gradient information while $\mathcal{M}_{\text{NASWOT}}$ 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.

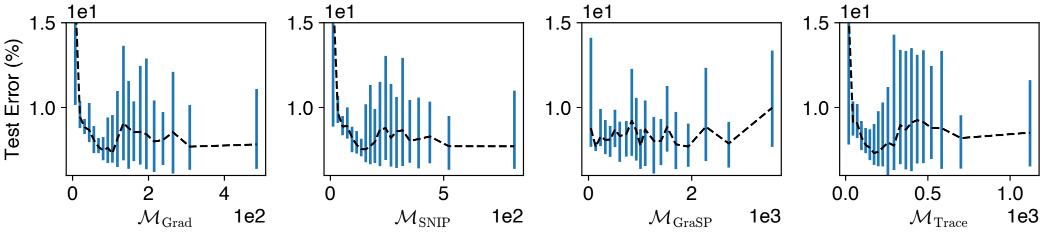
---

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

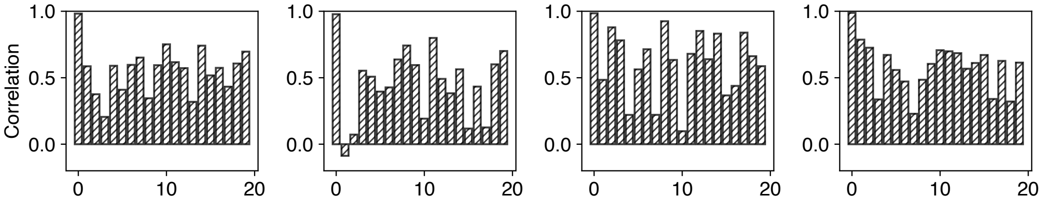
[3] Of note, the so-called gradient information contained in $\mathcal{M}_{\text{Fisher}}$ and $\mathcal{M}_{\text{SynFlow}}$ is different from the commonly used gradient of initialized model parameters that is derived from loss function or the output of DNN models. So, $\mathcal{M}_{\text{Fisher}}$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

Table 4: Connection between any two training-free metrics (i.e., $\mathcal{M}_1$ and $\mathcal{M}_2$ 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].

$\mathcal{M}_1$	$\mathcal{M}_2$	**NAS-Bench-101**			**NAS-Bench-201**		
		Pearson	Spearman	Kendall's Tau	Pearson	Spearman	Kendall's Tau
		Gradient-based training-free metrics					
$\mathcal{M}_{\text{Grad}}$	$\mathcal{M}_{\text{SNIP}}$	0.98	0.98	0.87	1.00	1.00	0.97
$\mathcal{M}_{\text{Grad}}$	$\mathcal{M}_{\text{GraSP}}$	0.35	0.61	0.43	0.60	0.92	0.77
$\mathcal{M}_{\text{Grad}}$	$\mathcal{M}_{\text{Trace}}$	0.98	0.98	0.87	0.98	0.97	0.85
$\mathcal{M}_{\text{SNIP}}$	$\mathcal{M}_{\text{GraSP}}$	0.34	0.59	0.42	0.55	0.92	0.77
$\mathcal{M}_{\text{SNIP}}$	$\mathcal{M}_{\text{Trace}}$	0.94	0.93	0.77	0.97	0.96	0.83
$\mathcal{M}_{\text{GraSP}}$	$\mathcal{M}_{\text{Trace}}$	0.37	0.57	0.40	0.69	0.89	0.73
$\mathcal{M}_{\text{KNAS}}$	$\mathcal{M}_{\text{Grad}}$	0.95	0.96	0.83	0.88	0.94	0.80
$\mathcal{M}_{\text{KNAS}}$	$\mathcal{M}_{\text{SNIP}}$	0.91	0.92	0.75	0.87	0.94	0.78
$\mathcal{M}_{\text{KNAS}}$	$\mathcal{M}_{\text{GraSP}}$	0.37	0.65	0.46	0.45	0.87	0.69
$\mathcal{M}_{\text{KNAS}}$	$\mathcal{M}_{\text{Trace}}$	0.96	0.96	0.84	0.89	0.97	0.86
		Non-gradient-based training-free metrics					
$\mathcal{M}_{\text{Fisher}}$	$\mathcal{M}_{\text{Trace}}$	0.69	0.97	0.85	0.30	0.78	0.69
$\mathcal{M}_{\text{SynFlow}}$	$\mathcal{M}_{\text{Trace}}$	0.02	0.50	0.34	0.07	0.49	0.35
$\mathcal{M}_{\text{NASWOT}}$	$\mathcal{M}_{\text{Trace}}$	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). Specifically, 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 κ applied in the following scenarios.

Dataset	**Scenario**	**Spearman**				**Kendall's Tau**			
		$\mathcal{M}_{\text{Grad}}$	$\mathcal{M}_{\text{SNIP}}$	$\mathcal{M}_{\text{GraSP}}$	$\mathcal{M}_{\text{Trace}}$	$\mathcal{M}_{\text{Grad}}$	$\mathcal{M}_{\text{SNIP}}$	$\mathcal{M}_{\text{GraSP}}$	$\mathcal{M}_{\text{Trace}}$
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 + κ	0.724	0.728	0.658	0.657	0.530	0.533	0.474	0.474
	Non-realizable	**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 + κ	0.716	0.719	0.649	0.651	0.527	0.529	0.469	0.470
	Non-realizable	**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 + κ	0.646	0.649	0.612	0.587	0.472	0.474	0.443	0.423
	Non-realizable	**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 $\mathcal{M}_{\text{opt}}$ for each training-free metric $\mathcal{M}$ that is capable of achieving the best generalization performance in the search space. When $\mathcal{M} < \mathcal{M}_{\text{opt}}$, architecture with a larger value of $\mathcal{M}$ typically enjoys a better generalization performance. On the contrary, when $\mathcal{M} > \mathcal{M}_{\text{opt}}$, architecture with a smaller value of $\mathcal{M}$ 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 [8] 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 κ 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 + κ as suggested by our Corollary 2. As revealed in Table 5, both trade-off and condition number κ 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 κ 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 κ 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**			
	$\mathcal{M}_{\text{Grad}}$	$\mathcal{M}_{\text{SNIP}}$	$\mathcal{M}_{\text{GraSP}}$	$\mathcal{M}_{\text{Trace}}$
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 five 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)	93.70±0.36	71.04±1.07	44.57±1.25	106
REINFORCE	93.85±0.37	71.71±1.09	45.24±1.18	103
HNAS ($\mathcal{M}_{\text{Grad}}$)	**94.04**±0.21	71.75±1.04	**45.91**±0.88	**20**
HNAS ($\mathcal{M}_{\text{SNIP}}$)	**93.94**±0.02	71.49±0.11	**46.07**±0.14	**20**
HNAS ($\mathcal{M}_{\text{GraSP}}$)	**94.13**±0.13	**72.59**±0.82	**46.24**±0.38	**20**
HNAS ($\mathcal{M}_{\text{Trace}}$)	**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 confirms the superior search efficiency and the remarkable search effectiveness of our HNAS framework.

C.5 HNAS in the DARTS Search Space

To support the effectiveness and efficiency of our HNAS, we also apply HNAS in the DARTS [5] search space to find well-performing architectures on CIFAR-10/100 and ImageNet [54]. Specifically, we sample a pool of 60000 architecture to evaluate their training-free metrics on CIFAR-10 in order to maintain high computational efficiency 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†	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†	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

† 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 $3\times10^{-4}$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 3×10^{-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 classifiers on ImageNet.

Algorithm	**Test Error (%)**		**Params** (M)	$+\times$ (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	-
ShuffleNet 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), [4] we wonder *whether our theoretical results are also applicable to DNNs using different initialization methods*, e.g., Xavier [70] and Kaiming [71] initialization. Specifically, 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 satisfies 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 defined 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 efficiency, 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 κ in this table. Specifically, in the upper part of Table 11, the correlations are evaluated using a batch size of 64 for κ and varying batch sizes for any training-free metric $\mathcal{M}$ from Sec. 3.2. Meanwhile, in the lower part of Table 11, the correlations are evaluated using varying batch sizes for κ and a batch size of 64 for any training-free metric $\mathcal{M}$. 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**			
	$\mathcal{M}_{\text{Grad}}$	$\mathcal{M}_{\text{SNIP}}$	$\mathcal{M}_{\text{GraSP}}$	$\mathcal{M}_{\text{Trace}}$	$\mathcal{M}_{\text{Grad}}$	$\mathcal{M}_{\text{SNIP}}$	$\mathcal{M}_{\text{GraSP}}$	$\mathcal{M}_{\text{Trace}}$
				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**			
	$\mathcal{M}_{\text{Grad}}$	$\mathcal{M}_{\text{SNIP}}$	$\mathcal{M}_{\text{GraSP}}$	$\mathcal{M}_{\text{Trace}}$	$\mathcal{M}_{\text{Grad}}$	$\mathcal{M}_{\text{SNIP}}$	$\mathcal{M}_{\text{GraSP}}$	$\mathcal{M}_{\text{Trace}}$
				Batch size 64 for κ and varying batch sizes for any $\mathcal{M}$				
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 κ and batch size 64 for any $\mathcal{M}$				
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 κ evaluated under varying batch sizes will lead to different results, indicating that κ is 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 κ 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 κ 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**			
	$\mathcal{M}_{\text{Grad}}$	$\mathcal{M}_{\text{SNIP}}$	$\mathcal{M}_{\text{GraSP}}$	$\mathcal{M}_{\text{Trace}}$	$\mathcal{M}_{\text{Grad}}$	$\mathcal{M}_{\text{SNIP}}$	$\mathcal{M}_{\text{GraSP}}$	$\mathcal{M}_{\text{Trace}}$
	4 channels for κ and varying channels for any $\mathcal{M}$							
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 κ and 32 channels for any $\mathcal{M}$							
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 $\mathcal{M}_{\text{KNAS}}$, $\mathcal{M}_{\text{Fisher}}$, $\mathcal{M}_{\text{SynFlow}}$ and $\mathcal{M}_{\text{NASWOT}}$ 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**			
	$\mathcal{M}_{\text{KNAS}}$	$\mathcal{M}_{\text{Fisher}}$	$\mathcal{M}_{\text{SynFlow}}$	$\mathcal{M}_{\text{NASWOT}}$	$\mathcal{M}_{\text{KNAS}}$	$\mathcal{M}_{\text{Fisher}}$	$\mathcal{M}_{\text{SynFlow}}$	$\mathcal{M}_{\text{NASWOT}}$
	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 κ 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 firstly 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 $\mathcal{M}_{\text{Fisher}}$, $\mathcal{M}_{\text{SynFlow}}$ and $\mathcal{M}_{\text{NASWOT}}$. Table 13 summarizes the results. Here, we use the same parameters applied in Sec. 6.2 for Corollary 2. While $\mathcal{M}_{\text{Fisher}}$ and $\mathcal{M}_{\text{SynFlow}}$ enjoy higher correlation to $\mathcal{M}_{\text{Trace}}$ than $\mathcal{M}_{\text{NASWOT}}$ in Appendix C.1, our generalization guarantees also performs better when using $\mathcal{M}_{\text{Fisher}}$ and $\mathcal{M}_{\text{SynFlow}}$. 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 $\mathcal{M}_{\text{Fisher}}$ and $\mathcal{M}_{\text{SynFlow}}$ can also find better-performing architectures than HNAS ($\mathcal{M}_{\text{NASWOT}}$). Surprisingly, HNAS ($\mathcal{M}_{\text{SynFlow}}$) 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**
	C10	C100	IN-16	(GPU Sec.)
HNAS ($\mathcal{M}_{\text{Grad}}$)	94.04±0.21	71.75±1.04	45.91±0.88	3010
HNAS ($\mathcal{M}_{\text{SNIP}}$)	93.94±0.02	71.49±0.11	46.07±0.14	2976
HNAS ($\mathcal{M}_{\text{GraSP}}$)	94.13±0.13	72.59±0.82	46.24±0.38	3148
HNAS ($\mathcal{M}_{\text{Trace}}$)	94.07±0.10	72.30±0.70	45.93±0.37	3006
HNAS ($\mathcal{M}_{\text{KNAS}}$)	94.19±0.06	72.94±0.52	46.31±0.38	3081
HNAS ($\mathcal{M}_{\text{Fisher}}$)	93.28±0.73	69.42±1.36	42.85±2.09	3309
HNAS ($\mathcal{M}_{\text{SynFlow}}$)	94.13±0.00	72.50±0.00	45.47±0.00	3615
HNAS ($\mathcal{M}_{\text{NASWOT}}$)	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
$\kappa/\mathcal{M}_{\text{Trace}}$	93.50	69.78	43.73
HNAS ($\mathcal{M}_{\text{Trace}}$)	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 $\mathcal{M}_{\text{Trace}}$. Note that in every BO iteration of Figure 5, the Spearman correlation we reported corresponds to the pair of hyperparameters μ and ν 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 μ and ν 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 μ and ν with prior knowledge. For example, by setting $\mu = 0$, we can obtain the training-free metric $\kappa/\mathcal{M}_{\text{Trace}}$. However, obtaining prior knowledge regarding the best choice of μ and ν for NAS is non-trivial. Therefore, tuning μ and ν 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 $\kappa/\mathcal{M}_{\text{Trace}}$ vs. our standard HNAS framework in Table 15. Notably, the results in Table 15 demonstrate that tuning μ and ν based on training-based performances can indeed lead to improved search results and therefore will be a better alternative than pre-defining μ and ν 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($\mathcal{M}_{\text{Trace}}$) and HNAS($\mathcal{M}_{\text{Grad}}$) 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.

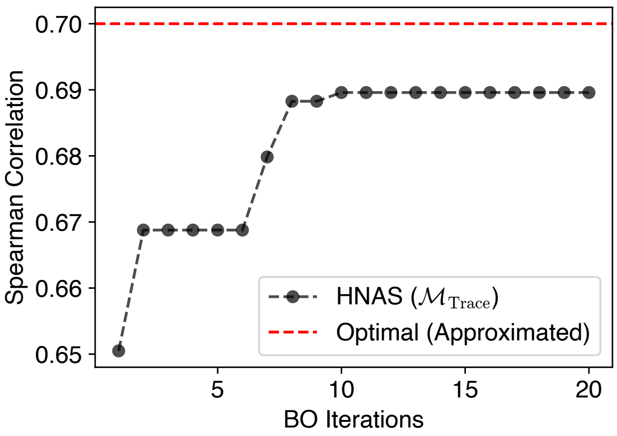


Figure 5: Evolution of the correlation between the test errors (on CIFAR-10) of all architectures in NAS-Bench-201 and their generalization guarantees (using $\mathcal{M}_{\text{Trace}}$) in the non-realizable scenario with the BO iterations in our HNAS framework.

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 ($\mathcal{M}_{\text{Grad}}$) w/ EI	94.04±0.21	71.75±1.04	45.91±0.88
HNAS ($\mathcal{M}_{\text{Grad}}$) w/ UCB	94.05±0.18	72.04±1.18	45.81±0.88
HNAS ($\mathcal{M}_{\text{Trace}}$) w/ EI	94.07±0.10	72.30±0.70	45.93±0.37
HNAS ($\mathcal{M}_{\text{Trace}}$) 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.