Neural Ensemble Search via Bayesian Sampling

Abstract

Recently, *neural architecture search* (NAS) has been applied to automate the design of neural networks in real-world applications. A large number of algorithms have been developed to improve the search cost or the performance of the final selected architectures in NAS. Unfortunately, these NAS algorithms aim to select only *one single* well-performing architecture from their search spaces and thus have overlooked the capability of *neural network ensemble* (i.e., an ensemble of neural networks with diverse architectures) in achieving improved performance over a single final selected architecture. To this end, we introduce a novel neural ensemble search algorithm, called *neural ensemble search via Bayesian sampling* (NESBS), to effectively and efficiently select well-performing neural network ensembles from a NAS search space. In our extensive experiments, NESBS algorithm is shown to be able to achieve improved performance over state-of-the-art NAS algorithms while incurring a comparable search cost, indicating the superior of our NESBS algorithm over these conventional NAS algorithms in practice.

1 INTRODUCTION

Recent years have witnessed a surging interest in designing well-performing architectures for different tasks. These architectures are typically manually designed by human experts, which requires numerous trials and errors during this manual design process and therefore is prohibitively costly. Consequently, the increasing demand for developing well-performing architectures in different tasks makes this manual design infeasible. To avoid such human efforts, Zoph and Le [2017] have introduced *neural architecture search* (NAS) to help automate the design of architectures. Since then, a number of NAS algorithms [Pham et al., 2018, Liu et al., 2019, Chen et al., 2019] have been developed to improve the search efficiency (i.e., the search cost) or the search effectiveness (i.e., the generalization performance of their final selected architectures) in NAS.

However, conventional NAS algorithms aim to select only *one single architecture* from their search spaces and hence have overlooked the capability of other candidate architectures from the same search spaces in helping improve the performance achieved by their final selected single architecture. That is, *neural network ensembles* are widely known to be capable of achieving an improved performance compared with a single neural network in practice [Gal and Ghahramani, 2016, Cortes et al., 2017, Lakshminarayanan et al., 2017]. This naturally begs the question: *How to select best-performing neural network ensembles with diverse architectures from a NAS search space in order to improve the performances achieved by existing NAS algorithms?* To the best of our knowledge, only limited efforts (e.g., [Zaidi et al., 2021]) have been devoted to this problem in the NAS literature. Unfortunately, the *neural ensemble search* (NES) algorithm based on random search or evolutionary algorithm in [Zaidi et al., 2021] requires excessive search costs to select their final neural network ensembles, which shall be unaffordable in resource-constrained scenarios.

To this end, this paper introduces a novel algorithm, namely *neural ensemble search via Bayesian sampling* (NESBS), to effectively and efficiently select the well-performing neural network ensemble with diverse architectures from a search space. We firstly represent the search space as a supernet following conventional one-shot NAS algorithms and then use the model parameters inherited from this supernet after its model training to estimate the single-model performances and also the ensemble performance of independently trained architectures (Sec. 3.1). Next, since both single-model performances and diverse model predictions affect the final ensemble performance according to [Zhou, 2012], we propose to use a variational posterior distribution of architectures based on a trained supernet to characterize these two factors,

Submitted to the 38th Conference on Uncertainty in Artificial Intelligence (UAI 2022). **To be used for reviewing only**.

i.e., single-model performances and diverse model predictions (Sec. 3.2). We then introduce two novel Bayesian sampling algorithms based on the posterior distribution of architectures, i.e., *Monte Carlo sampling* (MC Sampling) and *Stein Variational Gradient Descent with regularized diversity* (SVGD-RD), to effectively and efficiently select ensembles with both competitive single-model performances and compelling diverse model predictions (Sec. 3.3), which is also guaranteed to be able to achieve impressive ensemble performances [Zhou, 2012]. Lastly, we use extensive experiments to show that our NESBS algorithm is indeed able to select well-performing neural network ensembles effectively and efficiently in practice (Sec. 4).

2 RELATED WORKS & BACKGROUND

2.1 NEURAL ARCHITECTURE SEARCH

In the literature, many NAS algorithms [Zoph and Le, 2017, Zoph et al., 2018, Real et al., 2019] have been developed to automate the design of well-performing neural architectures. However, these NAS algorithms are inefficient in practice due to their requirement of the independent model training for each candidate architecture in the search space. To reduce such training costs, a supernet has been introduced to represent the search space and also share model parameters among the candidate architectures in the search space [Pham et al., 2018]. As a result, only the model training of this supernet is required, which can significantly improve the search efficiency of conventional NAS algorithms. After that, a number of one-shot NAS algorithms based on model parameter sharing [Chen et al., 2019, Liu et al., 2019, Xie et al., 2019, Chen and Hsieh, 2020, Chu et al., 2020] have been developed. Unfortunately, these algorithms aim to select *only one single architecture* from their search spaces. Thus, the capability of other candidate architectures from the same search spaces in helping improve the performance of their final selected single architecture have been overlooked.

2.2 NEURAL NETWORK ENSEMBLES

Meanwhile, neural network ensembles have been widely applied to improve the performance of a single neural network in different applications [Dietterich, 2000]. Over the years, a number of methods have been proposed to construct such neural network ensembles. For example, Gal and Ghahramani [2016] have proposed to use Monto Carlo Dropout to obtain neural network ensembles at test time. Meanwhile, *deep ensembles* (DeepEns) [Lakshminarayanan et al., 2017] adopt neural networks trained with different random initializations to construct ensembles and has achieved impressive performances on various tasks. Another line of ensemble works uses the checkpoints obtained during model training to build neural network ensembles [Huang et al., 2017]. More recently, Zaidi et al. [2021] have introduced *neural ensemble search* (NES) into NAS area to build well-performing neural network ensembles by selecting diverse architectures from the NAS search space, which has achieved competitive performance even compared with other ensemble methods. Unfortunately, the algorithm presented in [Zaidi et al., 2021] is shown to be prohibitively costly, which shall be unaffordable in resource-constrained scenarios. To this end, this paper presents a novel NESBS algorithm to advance this line of works (e.g., NES) by achieving state-of-the-art (SOTA) performances for neural network ensembles with diverse architectures while incurring a reduced search cost.

2.3 STEIN VARIATIONAL GRADIENT DESCENT

Stein Variational Gradient Descent (SVGD) [Liu and Wang, 2016] is a variational inference algorithm that approximates a target distribution $p(\boldsymbol{x})$ with a simpler density $q^*(\boldsymbol{x})$ in a predefined set $\mathcal{Q}$ by minimizing the *Kullback-Leibler* (KL) divergence between these two densities:

$$q^* = \arg\min_{q\in\mathcal{Q}}\{\mathrm{KL}(q||p) \triangleq \mathbb{E}_q\left[\log\left(q(\boldsymbol{x})/p(\boldsymbol{x})\right)\right]\} . \quad (1)$$

Specifically, SVGD represents $q^*(\boldsymbol{x})$ with a set of particles $\{\boldsymbol{x}_i\}_{i=1}^n$, which are firstly randomly initialized and then iteratively updated with updates $\boldsymbol{\phi}^*(\boldsymbol{x}_i)$ and a step size ϵ:

$$\boldsymbol{x}_i \leftarrow \boldsymbol{x}_i + \epsilon\boldsymbol{\phi}^*(\boldsymbol{x}_i), \quad \forall i \in \{1, \cdots, n\} . \quad (2)$$

Let $q_{[\epsilon\boldsymbol{\phi}]}$ denote the distribution of updated particles $\boldsymbol{x}' = \boldsymbol{x} + \epsilon\boldsymbol{\phi}(\boldsymbol{x})$. Let $\mathbb{F}$ denote the unit ball of a vector-valued reproducing kernel Hilbert space (RKHS) $\mathcal{H} \triangleq \mathcal{H}_0 \times \cdots \mathcal{H}_0$ where $\mathcal{H}_0$ is an RKHS formed by scalar-valued functions associated with a positive definite kernel $k(\boldsymbol{x}, \boldsymbol{x}')$. The work of [Liu and Wang, 2016] has shown that (2) can be viewed as functional gradient descent in the RKHS $\mathcal{H}$ and the optimal $\boldsymbol{\phi}^*$ in (2) can be obtained by solving the following problem

$$\boldsymbol{\phi}^* = \arg\max_{\boldsymbol{\phi}\in\mathbb{F}}\left\{-\frac{d}{d\epsilon}\mathrm{KL}(q_{[\epsilon\boldsymbol{\phi}]}||p)\Big|_{\epsilon=0}\right\} , \quad (3)$$

which yields a closed-form solution:

$$\boldsymbol{\phi}^*(\cdot) = \mathbb{E}_{\boldsymbol{x}\sim q}[k(\boldsymbol{x}, \cdot)\nabla_{\boldsymbol{x}}\log p(\boldsymbol{x}) + \nabla_{\boldsymbol{x}}k(\boldsymbol{x}, \cdot)] . \quad (4)$$

In practice, Liu and Wang [2016] have approximated the expectation in this close-form solution with the empirical mean of particles: $\boldsymbol{\phi}^*(\boldsymbol{x}_i) \approx \widehat{\boldsymbol{\phi}}^*(\boldsymbol{x}_i)$, where $\widehat{\boldsymbol{\phi}}^*(\boldsymbol{x}_i)$ is defined as

$$\widehat{\boldsymbol{\phi}}^*(\boldsymbol{x}_i) \triangleq \frac{1}{n}\sum_{j=1}^{n} k(\boldsymbol{x}_j, \boldsymbol{x}_i)\nabla_{\boldsymbol{x}_j}\log p(\boldsymbol{x}_j) + \nabla_{\boldsymbol{x}_j}k(\boldsymbol{x}_j, \boldsymbol{x}_i) . \quad (5)$$

As revealed in [Liu and Wang, 2016], the two terms in the aforementioned closed-form solution take different effects: The first term with $\nabla_{\boldsymbol{x}}\log p(\boldsymbol{x})$ favors particles with higher probability density, while the second term pushes the particles away from each other to encourage diversity.

3 NEURAL ENSEMBLE SEARCH VIA BAYESIAN SAMPLING

In contradiction to the selection of one single architecture in conventional NAS algorithm, this paper focuses on the problem of selecting a well-performing neural network ensemble with diverse architectures from the NAS search space, i.e., *neural ensemble search* (NES) [Zaidi et al., 2021]. Let $\boldsymbol{f}_{\mathcal{A}}(\boldsymbol{x}, \boldsymbol{\theta}_{\mathcal{A}})$ denote the output of an architecture $\mathcal{A}$ with input data $\boldsymbol{x}$ and model parameter $\boldsymbol{\theta}_{\mathcal{A}}$, let S be a set of architectures and $\boldsymbol{\Theta}_S$ be a set of the corresponding model parameters of these architectures. Given the ensemble scheme $\mathcal{F}_S(\boldsymbol{x}, \boldsymbol{\Theta}_S) \triangleq 1/n \sum_{\mathcal{A} \in S} \boldsymbol{f}_{\mathcal{A}}(\boldsymbol{x}, \boldsymbol{\theta}_{\mathcal{A}})$ with an ensemble size of $|S| = n$, [1] let $\mathcal{L}_{\text{train}}$ and $\mathcal{L}_{\text{val}}$ denote the training and validation loss respectively, NES can be formally framed as

$$\begin{aligned} &\min_{S} \mathcal{L}_{\text{val}}(\mathcal{F}_S(\boldsymbol{x}, \boldsymbol{\Theta}_S^*)) \\ &\text{s.t. } \forall\, \boldsymbol{\theta}_{\mathcal{A}}^* \in \boldsymbol{\Theta}_S^*,\ \boldsymbol{\theta}_{\mathcal{A}}^* = \arg\min_{\boldsymbol{\theta}_{\mathcal{A}}} \mathcal{L}_{\text{train}}(\boldsymbol{f}_{\mathcal{A}}(\boldsymbol{x}, \boldsymbol{\theta}_{\mathcal{A}}))\ . \end{aligned} \tag{6}$$

Unfortunately, (6) is challenging to solve mainly due to the following two reasons: (I) The enormous number of candidate architectures in the NAS search space, e.g., $\sim 10^{25}$ in the DARTS search space [Liu et al., 2019], makes the independent model training of every candidate architecture (i.e., the lower-level optimization in (6)) unaffordable. (II) The ensemble search space is exponentially increasing in the ensemble size n, e.g., there are $\sim m^n$ different ensembles given m diverse architectures. The combinatorial optimization problem (i.e., the upper-level optimization in (6)) is thus intractable to solve within this huge ensemble search space. Recently, Zaidi et al. [2021] have attempted to avoid these two problems by sampling a small pool of architectures from the search space for their final ensemble search. As a result, they fail to explore the whole search space and may achieve poor ensemble performances in practice. Moreover, their search cost is still unaffordable due to the independent model training of every architecture in the pool.

To this end, we novelly present the *neural ensemble search via Bayesian sampling* (NESBS) algorithm to solve (6) effectively and efficiently. We firstly employ the model parameters inherited from a supernet (i.e., a representation of the NAS search space) after its model training to estimate the single-model performances and also the ensemble performance of independently trained architectures (Sec. 3.1). This only requires the model training of the supernet and thus allows us to overcome the aforementioned challenge (I). We then derive a posterior distribution of architectures to characterize both the single-model performances and the diverse model predictions of candidate architectures in the search space (Sec. 3.2). Finally, based on this posterior distribution and also the aforementioned ensemble performance estimation, we introduce *Monte Carlo Sampling* (MC Sampling) and *Stein Variational Gradient Descent with regularized diversity* (SVGD-RD) to explore the ensembles in the whole search space effectively and efficiently (Sec. 3.3), which thus allows us to overcome the aforementioned challenge (II). An overview of our NESBS is in Algorithm 1.

[1] We apply this ensemble scheme for simplicity. Other ensemble schemes can also be used in the algorithm of this paper.

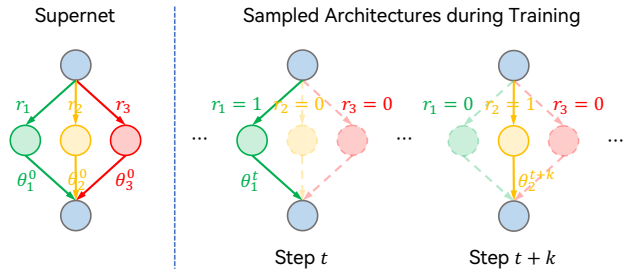


Figure 1: An illustration for the model training of supernet. Here, the supernet consists of three candidate architectures with r_i indicating the selection of one architecture and $\boldsymbol{\theta}_i^t$ denoting its model parameters at step t. In every training step, only one architecture is uniformly sampled to update its parameters and all other architectures will be ignored.

3.1 MODEL TRAINING OF SUPERNET

Similar to one-shot NAS algorithms [Pham et al., 2018, Liu et al., 2019], we represent NAS search space as a supernet. This then allows us to use the model parameters inherited from this trained supernet to estimate not only the single-model performances but also the ensemble performance of independently trained candidate architectures in the search space. However, in order to realize an accurate and fair estimation of these performances, we need to further ensure that every candidate architecture in the search space is trained for a comparable number of steps, namely the training fairness among candidate architectures [Chu et al., 2019]. To achieve this, in every training step of this supernet, we uniformly randomly sample one single candidate architecture from this supernet for model training (see Figure 1). The training fairness of such a training scheme can then be theoretically guaranteed, as demonstrated in our Appendix A. Moreover, we provide empirical results in Appendix C.1 to validate the effectiveness of such performance estimations.

3.2 DISTRIBUTION OF ARCHITECTURES

It has been demonstrated that both competitive single-model performances and diverse model predictions are required to achieve compelling ensemble performances [Zhou, 2012]. That is, NES algorithms should be capable of selecting architectures with both competitive single-model performances and diverse model predictions to achieve competitive ensemble performances. To realize this, we introduce a posterior distribution of architectures to firstly characterize these two factors. Let $\mathcal{D}$ denote the validation dataset, and let $p(\mathcal{A})$

and $p(\mathcal{A}|\mathcal{D})$ denote the prior and posterior distributions of a candidate architecture after its model training, where $p(\mathcal{A})$ follows from a categorical uniform distribution as required in Sec. 3.1. According to the Bayes' theorem, since $p(\mathcal{A})$ is uniform and $p(\mathcal{D})$ is constant,

$$p(\mathcal{A}|\mathcal{D}) = p(\mathcal{D}|\mathcal{A})p(\mathcal{A})/p(\mathcal{D}) \propto p(\mathcal{D}|\mathcal{A}) \tag{7}$$

where $p(\mathcal{D}|\mathcal{A})$ (i.e., the likelihood) is widely used to represent the single-model performance (i.e., loss) in practice. So, (7) implies that the posterior distribution $p(\mathcal{A}|\mathcal{D})$ can also characterize the single-model performances of architectures.

Meanwhile, given a γ-Lipschitz continuous loss function $\mathcal{L}(\boldsymbol{f})$, the diversity of model predictions, i.e., $\|\boldsymbol{f}_{\mathcal{A}_1} - \boldsymbol{f}_{\mathcal{A}_2}\|_2$, can then be lower bounded as below based on the Lipschitz continuity of $\mathcal{L}(\cdot)$,

$$\|\boldsymbol{f}_{\mathcal{A}_1} - \boldsymbol{f}_{\mathcal{A}_2}\|_2 \geq \gamma^{-1}|\mathcal{L}(\boldsymbol{f}_{\mathcal{A}_1}) - \mathcal{L}(\boldsymbol{f}_{\mathcal{A}_2})| \,. \tag{8}$$

Therefore, (8) suggests that in addition to being able to characterize the single-model performances of architectures (i.e., $\mathcal{L}(\boldsymbol{f})$), the posterior distribution $p(\mathcal{A}|\mathcal{D})$ can also estimate the diversity of model predictions for different architectures (e.g., $\mathcal{A}_1$ and $\mathcal{A}_2$) using $|p(\mathcal{A}_1|\mathcal{D}) - p(\mathcal{A}_2|\mathcal{D})|$.

However, it is intractable to obtain exact posterior distribution $p(\mathcal{A}|\mathcal{D})$ in the NAS search space. So, we approximate it with a variational distribution $p_{\boldsymbol{\alpha}}(\mathcal{A})$ (parameterized by a low-dimensional $\boldsymbol{\alpha}$) that can be optimized via variational inference, i.e., by minimizing the KL divergence between $p_{\boldsymbol{\alpha}}(\mathcal{A})$ and $p(\mathcal{A}|\mathcal{D})$. Equivalently, we only need to maximize a lower bound of the log-marginal likelihood, i.e., the *evidence lower bound* (ELBO) [Kingma and Welling, 2014], to get an optimal variational distribution $p_{\boldsymbol{\alpha}^*}(\mathcal{A})$:

$$\max_{\boldsymbol{\alpha}} \mathbb{E}_{\mathcal{A}\sim p_{\boldsymbol{\alpha}}(\mathcal{A})}\left[\log p(\mathcal{D}|\mathcal{A})\right] - \mathrm{KL}[p_{\boldsymbol{\alpha}}(\mathcal{A})||p(\mathcal{A})] \,, \tag{9}$$

Similar to [Kingma and Welling, 2014], gradient-based optimization algorithm with the reparameterization trick is employed to solve (9) efficiently (see Appendix B.3). While Xie et al. [2019] have adopted a similar form to (9) (without the KL term) *during* the model training of the supernet, namely the *best-response* posterior distribution, our *post-training* posterior distribution is able to not only provide a more accurate characterization of the single-model performances but also contribute to an improved ensemble search performance, as demonstrated in our Appendix C.2.

3.3 BAYESIAN SAMPLING

To solve (6) effectively and efficiently, we finally introduce two novel Bayesian sampling algorithms based on the posterior distribution of architectures in Sec. 3.2, i.e., *Monte Carlo sampling* (MC Sampling) and *Stein Variational Gradient Descent with regularized diversity* (SVGD-RD), to sample ensembles with both competitive single-model performances and compelling diversity of model predictions as required by well-performing ensembles [Zhou, 2012].

Algorithm 1 NES via Bayesian Sampling (NESBS)

Input: Iterations T, ensemble size n, a supernet
Train the supernet to get its tuned parameters $\boldsymbol{\theta}^*$
Obtain the posterior distribution $p_{\boldsymbol{\alpha}^*}(\mathcal{A})$ with (9)
for iteration $t = 1, \ldots, T$ **do**
  Sample S_t of size n via Algorithm 2 or 3
  Evaluate estimated $\mathcal{L}_{\text{val}}(\mathcal{F}_{S_t}(\boldsymbol{x}, \boldsymbol{\Theta}^*_{S_t}))$ given $\boldsymbol{\theta}^*$
end for
Select optimum $S^* = \arg\min_{S_t} \mathcal{L}_{\text{val}}(\mathcal{F}_{S_t}(\boldsymbol{x}, \boldsymbol{\Theta}^*_{S_t}))$

Algorithm 2 MC Sampling

Input: Ensemble size n, set $S = \varnothing$, posterior $p_{\boldsymbol{\alpha}^*}(\mathcal{A})$
for iteration $i = 1, \ldots, n$ **do**
  Sample $\mathcal{A}_i \sim p_{\boldsymbol{\alpha}^*}(\mathcal{A})$
  $S \leftarrow S \cup \{\mathcal{A}_i\}$
end for
Output: S

Algorithm 3 SVGD-RD

Input: Diversity coefficient δ, ensemble size n, iterations L, initial particles $\{\boldsymbol{x}_i^{(0)}\}_{i=1}^n$, posterior $p_{\boldsymbol{\alpha}^*}(\mathcal{A})$, kernel $k(\boldsymbol{x}, \boldsymbol{x}')$, step size $\{\epsilon_l\}_{l=1}^L$
for iteration $l = 0, \ldots, L-1$ **do**
  $\widehat{\boldsymbol{\phi}}_l^*(\boldsymbol{x}) = \frac{1}{n}\sum_{j=1}^n \nabla_{\boldsymbol{x}_j^{(l)}} k(\boldsymbol{x}_j^{(l)}, \boldsymbol{x}) - \delta\nabla_{\boldsymbol{x}} k(\boldsymbol{x}_j^{(l)}, \boldsymbol{x}) + k(\boldsymbol{x}_j^{(l)}, \boldsymbol{x})\nabla_{\boldsymbol{x}_j^{(l)}} \log p_{\boldsymbol{\alpha}^*}$
  $\boldsymbol{x}_i^{(l+1)} \leftarrow \boldsymbol{x}_i^{(l)} + \epsilon_l \widehat{\boldsymbol{\phi}}_l^*(\boldsymbol{x}_i^{(l)})$
end for
Output: $S = \{\mathcal{A}_i\}_{i=1}^n$ derived based on $\{\boldsymbol{x}_i^{(L)}\}_{i=1}^n$

3.3.1 Monte Carlo Sampling (MC Sampling)

Given the posterior distribution of architectures in Sec. 3.2, we firstly propose to use *Monte Carlo sampling* (MC Sampling) to sample a set of architectures from this posterior distribution (Algorithm 2). Note that MC Sampling guarantees that architectures with better single-model performances will be sampled (i.e., exploited) with higher probabilities, while architectures with diverse model predictions can also be sampled (i.e., explored) due to the inherent randomness in the sampling process. Compared with conventional NAS algorithms that select only one single well-performing architecture from the search space [Dong and Yang, 2019a, Xie et al., 2019], our MC sampling algorithm extends these algorithms by exploring the capability of diverse architectures while preserving its exploitation of architectures with compelling single-model performances.

3.3.2 SVGD with Regularized Diversity (SVGD-RD)

However, the diversity of sampled architectures using MC Sampling algorithm above is uncontrollable and hence may

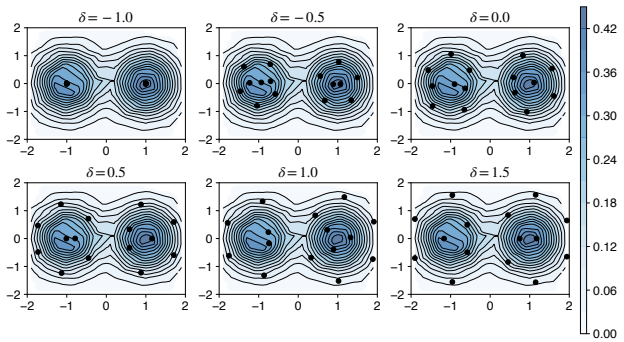


Figure 2: The impact of δ in SVGD-RD algorithm. We use contours and dots to denote the density of target distribution and sampled particles, respectively. The target distribution is chosen to be $p(\boldsymbol{x})=1/Z\,[\mathcal{N}(\boldsymbol{x}|\boldsymbol{u}_1,\Sigma_1)+\mathcal{N}(\boldsymbol{x}|\boldsymbol{u}_2,\Sigma_2)]$, where $\boldsymbol{u}_1=(-1,0)$, $\boldsymbol{u}_2=(0,1)$, $\Sigma_1=\Sigma_2=\mathrm{diag}((0.25,0.5))$ and Z denotes the normalization constant. Those sampled particles are obtained from our Algorithm 3 using L=1000, n=15, ϵ_l=0.1 and a *radial basis function* (RBF) kernel. Notably, SVGD-RD tends to sample particles with more diverse probability densities as δ is increased, indicating a controllable (via δ) diversity in our SVGD-RD algorithm. Meanwhile, SVGD-RD can consistently sample particles with high probability densities under varying δ.

lead to poor ensemble search results. So, in order to achieve a controllable diversity, we resort to *Stein Variational Gradient Descent* (SVGD). Theoretically, SVGD is capable of sampling particles with both large probability density and good diversity, where the diversity is explicitly encouraged (i.e., by the second term in (5)). Nonetheless, in practice, the particles sampled by SVGD may still fail to represent the target distribution well owing to the lack of diversity among those sampled particles as observed in [Zhuo et al., 2018]. Besides, the diversity of sampled particles in standard SVGD is still uncontrollable by human experts.

We hence develop *SVGD with regularized diversity* (SVGD-RD) sampling algorithm, which is amenable to achieve a controllable diversity among those sampled particles. Here, we follow the notations from Sec. 2.3. In particular, when optimizing the distribution q^* (represented by the n particles $\{\boldsymbol{x}_i^*\}_{i=1}^n$), we modify the objective in (1) by adding a term representing the (controllable) diversity among the particles measured by the kernel function $k(\boldsymbol{x},\boldsymbol{x}')$:

$$q^* = \arg\min_{q\in\mathcal{Q}} \mathrm{KL}(q||p) + n\delta\mathbb{E}_{\boldsymbol{x},\boldsymbol{x}'\sim q}\left[k(\boldsymbol{x},\boldsymbol{x}')\right] \ , \quad (10)$$

where δ is the parameter explicitly controlling the diversity. Here, p in (10) denotes the posterior distribution $p_{\boldsymbol{\alpha}^*}(\mathcal{A})$ derived in Sec. 3.2 which we intend to sample from. Following the work of SVGD, q^* in (10) is represented by $\{\boldsymbol{x}_i^*\}_{i=1}^n$, denoting our final selected neural network ensemble that can achieve both competitive single-model performances (i.e., large probability density) and also diverse model predictions.

Our Proposition 1 below provides one possible update rule for the particles $\{\boldsymbol{x}_i\}_{i=1}^n$ to optimize (10) (see its proof in Appendix A). Finally, Algorithm 3 summarizes the details of our SVGD-RD algorithm and Appendix B.4 provides its optimization details in practice. After obtaining those optimal particles $\{\boldsymbol{x}_i^*\}_{i=1}^n$ in our SVGD-RD algorithm, we then apply these particles to derive the architectures in our final selected ensembles (see details in Appendix B.4).

Proposition 1. *Given the proximal operator* $\mathrm{prox}_h(\boldsymbol{y}) = \arg\min_{\boldsymbol{z}} h(\boldsymbol{z}) + 1/2\|\boldsymbol{z}-\boldsymbol{y}\|_2^2$*, by applying proximal gradient method [Parikh and Boyd, 2014] and proper approximation,* (10) *can be optimized via the following updates for the particles* $\{\boldsymbol{x}_i\}_{i=1}^n$*,*

$$\boldsymbol{x}_i \leftarrow \boldsymbol{x}_i + 1/n\textstyle\sum_{j=1}^n k(\boldsymbol{x}_j,\boldsymbol{x}_i)\nabla_{\boldsymbol{x}_j}\log p(\boldsymbol{x}_j) + \nabla_{\boldsymbol{x}_j}k(\boldsymbol{x}_j,\boldsymbol{x}_i) - \delta\nabla_{\boldsymbol{x}_i}k(\boldsymbol{x}_j,\boldsymbol{x}_i)\ .$$

Compared with MC Sampling, our SVGD-RD algorithm provides a controllable trade-off between the single-model performances and the diverse model predictions. On the one hand, the minimization of the KL divergence term in (10) encourages the selection of architectures with competitive single-model performances by favoring particles with high probability densities, as shown by Proposition 2 below (proof given in Appendix A).[2] On the other hand, the maximization of the scaled distance $-n\delta\mathbb{E}_{\boldsymbol{x},\boldsymbol{x}'\sim q}\left[k(\boldsymbol{x},\boldsymbol{x}')\right]$ among the sampled particles leads to a controllable diversity (via δ) among these sampled particles and also a controllable diversity of the probability densities among these particles (see Figure 2), which also implies a controllable diversity of the model predictions as suggested in Sec. 3.2.

Proposition 2. *Let* p *be a target density and* $k(\boldsymbol{x},\boldsymbol{x}')=c$ *for every* $\boldsymbol{x}=\boldsymbol{x}'$ *where* c *is a constant. For any* $\delta\in\mathbb{R}$*, our SVGD-RD algorithm is equivalent to the maximization of the density* p *w.r.t* $\boldsymbol{x}$ *in the case of* $n=1$*.*

4 EXPERIMENTS

4.1 SEARCH IN NAS-BENCH-201

To verify the effectiveness and efficiency of NESBS algorithm, we firstly compare it with other well-known NAS and ensemble (search) algorithms in NAS-Bench-201 [Dong and Yang, 2020]. Table 1 summarizes the results. Table 1 shows that ensemble (search) algorithms, including our NESBS, consistently achieve improved generalization performance over conventional NAS algorithms. This is because ensemble (search) algorithms will select neural network ensembles whereas NAS algorithms will select only one single architecture. Moreover, it has been widely verified that model

[2] Although Proposition 2 is only applicable in the case of $n=1$, our SVGD-RD is still capable of sampling particles with high probability densities when n>1, as validated in Figure 2.

Table 1: The comparison of architectures selected by different NAS and ensemble (search) algorithms in NAS-Bench-201 with ensemble size n=3. The test errors are reported with the mean and standard error of three independent trials and our search costs are evaluated on a single Nvidia 1080Ti GPU. The results marked by † are reported by Dong and Yang [2020].

Architecture(s)	**Test Error** (%)			**Search Cost** (GPU Hours)
	CIFAR-10	**CIFAR-100**	**ImageNet-16-200**	
		Manual design		
ResNet† [He et al., 2016]	6.03	29.14	56.37	-
		NAS algorithms		
ENAS† [Pham et al., 2018]	45.70±0.00	84.39±0.00	83.68±0.00	3.7
DARTS† (2nd) [Liu et al., 2019]	45.70±0.00	84.39±0.00	83.68±0.00	8.3
GDAS† [Dong and Yang, 2019a]	6.49±0.13	29.39±0.26	58.16±0.90	8.0
SETN† [Dong and Yang, 2019b]	13.81±4.63	43.13±7.77	68.10 ±4.07	8.6
RSPS† [Li and Talwalkar, 2019]	12.34±1.69	41.67±4.34	68.86±3.88	2.1
		Ensemble (search) algorithms		
DeepEns [Lakshminarayanan et al., 2017]	5.75	25.27	54.70	-
NES-RS [Zaidi et al., 2021]	5.83±0.33	25.58±0.84	54.34±1.67	5.1
		Our ensemble search algorithm		
NESBS (MC Sampling)	5.76±0.25	25.39±0.69	**53.47**±1.75	**1.1**
NESBS (SVGD-RD)	5.92±0.07	**25.00**±0.17	**52.68**±0.35	**1.2**

ensembles generally outperform a single machine learning model in practice [Zhou, 2012]. In addition, our NESBS algorithm outperforms other ensemble (search) baseline (i.e., DeepEns and NES-RS) especially on large-scale datasets (i.e., CIFAR-100 [Krizhevsky et al., 2009] and ImageNet-16-200 [Chrabaszcz et al., 2017]) while incurring less search costs than NES-RS, which thus implies the superior of our NESBS over these ensemble (search) baselines. Even on a small-scale dataset (i.e., CIFAR-10), our NESBS can also achieve comparable search results to DeepEns and NES-RS. More interestingly, our NESBS algorithm is even able to incur reduced search costs than conventional NAS algorithms. This is likely because more training epochs have been used in these NAS algorithms, whereas a small number of training epochs can already contribute to well-performing results for our NESBS algorithm.

4.2 SEARCH IN THE DARTS SEARCH SPACE

We further demonstrate the superior search effectiveness and efficiency of our NESBS by comparing it with other NAS and ensemble (search) baselines in a larger search space, i.e., the DARTS [Liu et al., 2019] search space, using both classification and adversarial defense tasks on CIFAR-10/100 or ImageNet [Deng et al., 2009]. We follow Appendix B.5 to evaluate the final neural network ensembles selected by our NESBS algorithm with ensemble size n=3, T=5 and optimization details in Appendix B.

Ensemble for classification. Table 2 summarizes the comparison of classification performances on CIFAR-10/100. Similar to the results in Sec. 4.1, ensemble (search) algorithms, including our NESBS, are generally able to achieve improved generalization performances over conventional NAS algorithms, which thus justifies the essence of ensemble (search) algorithms for improved performance. Notably, even compared with other ensemble baselines such as MC DropPath (i.e., developed following Monte Carlo Dropout [Gal and Ghahramani, 2016]) and DeepEns, our NESBS is still able to achieve improved performances. Since these ensemble baselines are orthogonal to our NESBS, they can be integrated into our NESBS for further performance improvement in real-world applications. More importantly, our algorithm outperforms NES-RS, by achieving both improved search effectiveness (lowest test errors) and efficiency (lowest search costs). Furthermore, our NESBS even incurs comparable search costs compared with the most efficient NAS algorithms (e.g., GDAS, P-DARTS), which also highlights the efficiency of our NESBS. Similar results on ImageNet can be achieved by our NESBS as shown in Table 3. [3]

Ensemble for adversarial defense. Ensemble methods have already been shown to be an essential and effective defense mechanism against adversarial attacks [Strauss et al., 2017]. Specifically, an adversarial attacker can only use *a single model* randomly sampled from an ensemble to generate the adversarial examples, whereas the ensemble method defends against adversarial attacks (i.e., makes its predictions) using *all models* in this ensemble. Ensemble methods

[3] Following the convention of NAS and ensemble search algorithms in Table 3, the ensembles selected by our NESBS are also searched on CIFAR-10 and then transferred to ImageNet.

Table 2: The comparison of different image classifiers on CIFAR-10/100. The results of MC DropPath are from a drop probability of 0.01 and our search costs are evaluated on Nvidia 1080Ti.

Architecture(s)	**Test Error** (%)		**Params** (M)		**Search Cost** (GPU Days)	**Search Method**
	C10	**C100**	**C10**	**C100**		
			NAS algorithms			
NASNet-A [Zoph et al., 2018]	2.65	-	3.3	-	2000	RL
AmoebaNet-A [Real et al., 2019]	3.34	18.93	3.2	3.1	3150	evolution
PNAS [Liu et al., 2018]	3.41	19.53	3.2	3.2	225	SMBO
ENAS [Pham et al., 2018]	2.89	19.43	4.6	4.6	0.5	RL
DARTS [Liu et al., 2019]	2.76	17.54	3.3	3.4	1	gradient
GDAS [Dong and Yang, 2019a]	2.93	18.38	3.4	3.4	0.3	gradient
P-DARTS [Chen et al., 2019]	2.50	-	3.4	-	0.3	gradient
DARTS- (avg) [Chu et al., 2020]	2.59	17.51	3.5	3.3	0.4	gradient
SDARTS-ADV [Chen and Hsieh, 2020]	2.61	-	3.3	-	1.3	gradient
			Ensemble (search) algorithms			
MC DropPath (ENAS)	2.88	16.83	3.8‡	3.9‡	-	-
DeepEns (ENAS)	2.49	15.04	3.8‡	3.9‡	-	-
DeepEns (DARTS)	2.42	14.56	3.3‡	3.4‡	-	-
NES-RS♯ [Zaidi et al., 2021]	2.50	15.24	3.0‡	3.1‡	0.7	greedy
			Our ensemble search algorithm			
NESBS (MC Sampling)	**2.41**	14.70	3.8‡	3.9‡	**0.2**	sampling
NESBS (SVGD-RD)	**2.36**	**14.55**	3.7‡	3.8‡	**0.2**	sampling

‡ Reported as the averaged parameter size of the architectures in a neural network ensemble.
♯ Obtained from a pool of size 50, in which every architecture is uniformly randomly sampled from the DARTS search spaces and then trained independently for 50 epochs following the evaluation settings in Appendix B.5.

Table 3: The comparison of image classifiers on ImageNet. The ensemble size is set to n=3 for NES-RS and NESBS.

Architecture(s)	**Test Error** (%)		**Params** (M)	$+\times$ (M)
	Top-1	**Top-5**		
NAS algorithms				
NASNet-A	26.0	8.4	5.3	564
AmoebaNet-A	25.5	8.0	5.1	555
PNAS	25.8	8.1	5.1	588
DARTS	26.7	8.7	4.7	574
GDAS	26.0	8.5	5.3	581
P-DARTS	24.4	7.4	4.9	557
SDARTS-ADV	25.2	7.8	5.4	594
Ensemble search algorithms				
NES-RS	23.4	6.8	3.9	432
Our ensemble search algorithm				
NESBS (MC Sampling)	**22.3**	**6.2**	4.6	522
NESBS (SVGD-RD)	**22.3**	**6.1**	4.9	562

can defend against the adversarial attacks in such a setting because the generated adversarial examples using only one single model are unlikely to fool all models in an ensemble. More experimental details are in Appendix B.6. Our results on adversarial defense task are summarized in Table 4. Table 4 shows that ensemble (search) algorithms are indeed able to significantly improve the performance of adversarial defense, i.e., the test accuracies in the defense column are consistently higher than the ones in attack column. More importantly, even under different white-box adversarial attacks, our NESBS algorithm can generally achieve improved defense performances (i.e., higher test accuracy in the *Defense* columns) than other baselines including DeepEns and NES-RS. These results thus further support the effectiveness of our NESBS over existing ensemble (search) algorithms.

4.3 SINGLE-MODEL PERFORMANCES & DIVERSITY OF MODEL PREDICTIONS

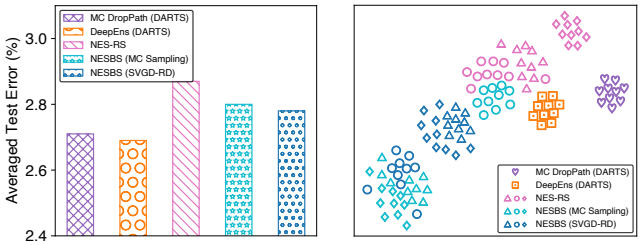
We demonstrate that the effectiveness of our NESBS result from its ability to achieve a good trade-off between the single-model performances and the diversity of model predictions. We firstly quantitatively compare the single-model performances (measured by the *averaged test error* (ATE) of the models in an ensemble) and the diversity of model predictions (measured by the *pairwise predictive disagreement* (PPD) of an ensemble [Fort et al., 2019]) achieved by different ensemble (search) algorithms on CIFAR-10/100. We further qualitatively visualize their single-model performances and diverse model predictions using a histogram of the averaged test error (ATE) of the models in their ensem-

Table 4: The comparison of adversarial defense among different ensemble (search) algorithms on CIFAR-10/100 under white-box adversarial attacks. The *Attack* and *Defense* columns denote the test *accuracy* under the attack using a single model randomly sampled from an ensemble and the defense using the whole ensemble, respectively. Each result reports the mean and standard deviation of test accuracies for 3 rounds of the attack-defense process with an ensemble size of $n = 3$.

Dataset	Method	FGSM		PGD-40		CW	
		Attack (%)	Defense (%)	Attack (%)	Defense (%)	Attack (%)	Defense (%)
C10	DeepEns	-	-	-	-	-	-
	↪ ENAS	77.85±0.58	87.94±0.21	59.51±1.13	86.57±0.15	31.36±1.20	85.20±0.77
	↪ DARTS	76.79±0.80	88.21±0.14	57.71±1.65	82.02±0.10	26.90±1.37	82.46±0.35
	NES-RS	79.19±1.39	89.32±0.27	65.59±2.11	85.22±0.41	37.20±4.62	86.75±0.88
	NESBS (MC Sampling)	78.75±1.29	89.15±0.08	63.60±1.87	85.35±0.31	**37.71**±1.97	**86.86**±0.66
	NESBS (SVGD-RD)	79.12±0.61	**89.86**±0.33	65.53±1.56	85.37±0.38	**38.27**±1.27	86.00±1.10
C100	DeepEns	-	-	-	-	-	-
	↪ ENAS	46.40±0.37	64.94±0.27	28.87±0.27	56.79±0.25	9.60±0.30	69.43±0.44
	↪ DARTS	46.98±0.57	65.38±0.23	28.78±0.74	57.10±0.04	9.73±0.43	70.15±0.29
	NES-RS	47.10±1.46	65.33±0.36	30.68±1.66	58.80±0.80	9.96±1.45	70.24±0.33
	NESBS (MC Sampling)	**50.69**±1.58	**67.63**±0.05	**33.37**±0.42	**60.36**±0.62	**15.64**±2.83	**71.25**±1.27
	NESBS (SVGD-RD)	**51.47**±0.40	**66.66**±0.13	**35.02**±0.37	**59.96**±0.18	**16.72**±0.61	**69.88**±0.16

Table 5: The quantitative comparison of the single-model performances (measured by ATE (%), smaller is better) and the diversity of model predictions (measured by PPD (%), larger is better) achieved by different ensemble (search) algorithms with an ensemble size of 3 on CIFAR-10/100.

Method	C10		C100	
	ATE	PPD	ATE	PPD
MC DropPath (DARTS)	2.71	0.39	16.68	2.63
DeepEns (DARTS)	**2.69**	2.08	**16.18**	12.45
NES-RS	2.87	2.29	17.20	**14.14**
NESBS (MC Sampling)	2.80	**2.57**	16.70	13.84
NESBS (SVGD-RD)	2.78	2.27	16.50	13.16



(a) Single-model performances (b) Diverse model predictions

Figure 3: The qualitative comparison of (a) the single-model performances and (b) the diverse model predictions achieved by different ensemble (search) algorithms with an ensemble size of $n = 3$ on CIFAR-10. Each architecture in (b) is independently evaluated for ten times to visualize their model predictions, which follows from DeepEns.

bles and a t-SNE [Van der Maaten and Hinton, 2008] plot of their model predictions, respectively.

Table 5 and Figure 3 present the results of our quantitative and qualitative comparisons, respectively. Compared with the ensemble baselines of MC DropPath and DeepEns, our NESBS is capable of enjoying a larger diversity of model predictions while preserving competitive single-model performances. Meanwhile, compared with the ensemble search baselines of NES-RS, our algorithm can achieve improved single-model performances while maintaining comparably diverse model predictions. These results suggest that our NESBS is able to select ensembles achieving a better trade-off between the single-model performances and the diversity of model predictions among these baselines, which is known to be an important criterion for well-performing ensembles [Zhou, 2012]. As a result, Table 5 and Figure 3 provide empirical justifications for the improved effectiveness achieved by our NESBS.

5 CONCLUSION

This paper presents a novel neural ensemble search algorithms, called NESBS, that can effectively and efficiently select well-performing neural network ensembles with diverse architectures from a NAS search space. Our extensive experiments have shown that NESBS is able to achieve improved performances while preserving a comparable search cost compared with conventional NAS algorithms. Moreover, even compared with other ensemble (search) baselines (e.g., DeepEns and NES-RS), our NESBS is also capable of enjoying boosted search effectiveness and efficiency, which further suggests the superior of our NESBS in practice.

References

Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. In *Proc. IEEE S&P*, pages 39–57, 2017.

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

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

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

Xiangxiang Chu, Bo Zhang, Ruijun Xu, and Jixiang Li. FairNAS: Rethinking evaluation fairness of weight sharing neural architecture search. arXiv:1907.01845, 2019.

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.

Corinna Cortes, Xavier Gonzalvo, Vitaly Kuznetsov, Mehryar Mohri, and Scott Yang. Adanet: Adaptive structural learning of artificial neural networks. In *Proc. ICML*, pages 874–883, 2017.

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.

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

Thomas G. Dietterich. Ensemble methods in machine learning. In *Multiple Classifier Systems*, pages 1–15, 2000.

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

Xuanyi Dong and Yi Yang. One-shot neural architecture search via self-evaluated template network. In *Proc. ICCV*, pages 3680–3689, 2019b.

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

Stanislav Fort, Huiyi Hu, and Balaji Lakshminarayanan. Deep ensembles: A loss landscape perspective. arXiv:1912.02757, 2019.

Yarin Gal and Zoubin Ghahramani. Dropout as a Bayesian approximation: Representing model uncertainty in deep learning. In *Proc. ICML*, pages 1050–1059, 2016.

Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In *Proc. ICLR*, 2015.

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

Weijun Hong, Guilin Li, Weinan Zhang, Ruiming Tang, Yunhe Wang, Zhenguo Li, and Yong Yu. DropNAS: Grouped operation dropout for differentiable architecture search. In *Proc. IJCAI*, pages 2326–2332, 2020.

Gao Huang, Yixuan Li, Geoff Pleiss, Zhuang Liu, John E. Hopcroft, and Kilian Q. Weinberger. Snapshot ensembles: Train 1, get M for free. In *Proc. ICLR*, 2017.

Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with Gumbel-Softmax. In *Proc. ICLR*, 2017.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *Proc. ICLR*, 2015.

Diederik P. Kingma and Max Welling. Auto-encoding variational Bayes. In *Proc. ICLR*, 2014.

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

Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. In *Proc. NIPS*, pages 6402–6413, 2017.

Liam Li and Ameet Talwalkar. Random search and reproducibility for neural architecture search. In *Proc. UAI*, pages 367–377, 2019.

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

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

Qiang Liu and Dilin Wang. Stein variational gradient descent: A general purpose bayesian inference algorithm. In *Proc. NIPS*, pages 2370–2378, 2016.

Chris J. Maddison, Andriy Mnih, and Yee Whye Teh. The Concrete distribution: A continuous relaxation of discrete random variables. In *Proc. ICLR*, 2017.

Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. In *Proc. ICLR*, 2018.

Neal Parikh and Stephen P. Boyd. Proximal algorithms. *Found. Trends Optim.*, 1(3):127–239, 2014.

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

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

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

Thilo Strauss, Markus Hanselmann, Andrej Junginger, and Holger Ulmer. Ensemble methods as a defense to adversarial perturbations against deep neural networks. arXiv:1709.03423, 2017.

Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-SNE. *J. Mach. Learn. Res.*, 9(11), 2008.

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

Yuhui Xu, Lingxi Xie, Xiaopeng Zhang, Xin Chen, Guo-Jun Qi, Qi Tian, and Hongkai Xiong. PC-DARTS: Partial channel connections for memory-efficient architecture search. In *Proc. ICLR*, 2020.

Sheheryar Zaidi, Arber Zela, Thomas Elsken, Chris C Holmes, Frank Hutter, and Yee Teh. Neural ensemble search for uncertainty estimation and dataset shift. In *Proc. NeurIPS*, 2021.

Zhi-Hua Zhou. *Ensemble methods: Foundations and algorithms*. CRC press, 2012.

Jingwei Zhuo, Chang Liu, Jiaxin Shi, Jun Zhu, Ning Chen, and Bo Zhang. Message passing stein variational gradient descent. In *Proc. ICML*, pages 6013–6022, 2018.

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

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

A PROOFS

Proposition. (Training fairness in the supernet.) *Let T and $T_{\mathcal{A}}$ denote the number of steps applied to train the supernet and candidate architecture $\mathcal{A}$ in the search space of size N, by uniformly randomly sampling a single architecture from this search space for the model training in each step, we have*

$$\Pr(\lim_{T\to\infty} T_{\mathcal{A}_i}/T = \lim_{T\to\infty} T_{\mathcal{A}_j}/T) = 1 \quad \forall i,j \in \{1,\cdots,N\} \,.$$

Proof. Let random variable $X_i^t \in \{0,1\}$ denote the selection of candidate architecture $\mathcal{A}_i$ at training step t under our sampling scheme in the proposition above. For any $t > 0$ and $i,j \in [N]$, random variable $X_i^t - X_j^t$ can achieve following possible assignments and probabilities (denoted by p):

$$X_i^t - X_j^t = \begin{cases} +1, & p = 1/N \\ 0, & p = (N-2)/N \\ -1, & p = 1/N \end{cases} \,. \tag{11}$$

Consequently, $\mathbb{E}[X_i^t - X_j^t] = 0$. According to the strong law of large numbers, we further have

$$\Pr(\lim_{T\to\infty} T^{-1} \sum_{t=1}^{T} X_i^t - X_j^t = 0) = 1 \,. \tag{12}$$

Note that

$$T^{-1}(T_{\mathcal{A}_i} - T_{\mathcal{A}_j}) = T^{-1} \sum_{t=1}^{T} X_i^t - X_j^t \,. \tag{13}$$

We thus can complete this proof by

$$\Pr(\lim_{T\to\infty} T^{-1}(T_{\mathcal{A}_i} - T_{\mathcal{A}_j}) = 0) = 1 \,. \tag{14}$$

□

Proof of Proposition 1. As particles $\{\boldsymbol{x}_i\}_{i=1}^n$ of size n are applied to approximate the density q in our SVGD-RD, the second term (i.e., the controllable diversity term) in our (10) can then be approximated using these particles as

$$n\delta\mathbb{E}_{\boldsymbol{x},\boldsymbol{x}'\sim q}\left[k(\boldsymbol{x},\boldsymbol{x}')\right] \approx \delta/n \sum_{i=1}^{n}\sum_{j=1}^{n} k(\boldsymbol{x}_i,\boldsymbol{x}_j) \triangleq \sum_{i=1}^{n} h(\boldsymbol{x}_i) \,, \tag{15}$$

where $h(\boldsymbol{x}) \triangleq \delta/n \sum_{j=1}^{n} k(\boldsymbol{x},\boldsymbol{x}_j)$. We take $\boldsymbol{x}_j$ in $k(\boldsymbol{x},\boldsymbol{x}_j)$ as a constant for the approximation above. Consequently, we have

$$\nabla_{\boldsymbol{x}_k} \sum_{i=1}^{n} h(\boldsymbol{x}_i) = \nabla_{\boldsymbol{x}_k} h(\boldsymbol{x}_k) \,. \tag{16}$$

Let $\boldsymbol{x}_i^+ \triangleq \boldsymbol{x}_i + \epsilon\boldsymbol{\phi}^*(\boldsymbol{x}_i)$ ($\forall i \in \{1,\cdots,n\}$) denote the functional gradient decent in the RKHS $\mathcal{H}$ to minimize the

KL divergence term in our (10). Based on (16) above, given proximal operator $\text{prox}_h(\boldsymbol{x}^+) = \arg\min_{\boldsymbol{y}} h(\boldsymbol{y}) + 1/2\|\boldsymbol{y} - \boldsymbol{x}^+\|_2^2$, by using proximal gradient method Parikh and Boyd [2014], our (10) can then be optimized via the following update to each particle $\boldsymbol{x}_i$:

$$\boldsymbol{x}_i \leftarrow \text{prox}_h(\boldsymbol{x}_i^+) = \arg\min_{\boldsymbol{y}} h(\boldsymbol{y}) + 1/2\|\boldsymbol{y} - \boldsymbol{x}_i^+\|_2^2 \,. \quad (17)$$

According to the *Karush-Kuhn-Tucker* (KKT) conditions, the local optimum $\boldsymbol{y}^*$ of this proximal operator satisfies

$$\text{prox}_h(\boldsymbol{x}_i^+) = \boldsymbol{y}^* = \boldsymbol{x}_i^+ - \nabla_{\boldsymbol{y}^*} h(\boldsymbol{y}^*) \,. \quad (18)$$

When $h(\cdot)$ is convex, this local optimum is also a global optimum. As (16) is intractable to solve given a complex $h(\cdot)$, we approximate $h(\boldsymbol{y}^*)$ with its first-order Taylor expansion, i.e., $h(\boldsymbol{y}^*) \approx h(\boldsymbol{x}_i) + \nabla_{\boldsymbol{x}_i} h(\boldsymbol{x}_i)(\boldsymbol{y}^* - \boldsymbol{x}_i)$ and achieve following approximation:

$$\begin{aligned} \text{prox}_h(\boldsymbol{x}_i^+) &\approx \boldsymbol{x}_i^+ - \nabla_{\boldsymbol{x}_i} h(\boldsymbol{x}_) \\ &\approx \boldsymbol{x}_i + \epsilon \boldsymbol{\phi}^*(\boldsymbol{x}_i) - \nabla_{\boldsymbol{x}_i} h(\boldsymbol{x}_i) \\ &\approx \boldsymbol{x}_i + \epsilon \boldsymbol{\phi}^*(\boldsymbol{x}_i) - \delta/n \textstyle\sum_j^n \nabla_{\boldsymbol{x}_i} k(\boldsymbol{x}_i, \boldsymbol{x}_j) \,. \end{aligned} \quad (19)$$

Given the approximation $\boldsymbol{\phi}^*(\boldsymbol{x}_i) \approx \widehat{\boldsymbol{\phi}}^*(\boldsymbol{x}_i)$ and the definition of $\widehat{\boldsymbol{\phi}}^*(\boldsymbol{x}_i)$ in (5), we complete our proof by

$$\begin{aligned} \boldsymbol{x}_i \leftarrow \boldsymbol{x}_i &+ 1/n \textstyle\sum_{j=1}^n k(\boldsymbol{x}_j, \boldsymbol{x}_i) \nabla_{\boldsymbol{x}_j} \log p(\boldsymbol{x}_j) \\ &+ \nabla_{\boldsymbol{x}_j} k(\boldsymbol{x}_j, \boldsymbol{x}_i) - \delta \nabla_{\boldsymbol{x}_i} k(\boldsymbol{x}_j, \boldsymbol{x}_i) \,. \end{aligned} \quad (20)$$

Proof of Proposition 2. Notably, since $k(\boldsymbol{x}, \boldsymbol{x}') = c$ when $\boldsymbol{x} = \boldsymbol{x}'$, we will achieve a constant $k(\boldsymbol{x}, \boldsymbol{x})$ for any particle $\boldsymbol{x}$ in the case of $n = 1$, which can be ignored in our SVGD-RD for any $\delta \in \mathbb{R}$. In light of this, our SVGD-RD in the case of $n = 1$ degenerates into standard SVGD. Consequently, to prove Proposition 2, we only need to consider SVGD in the case of $n = 1$.

Considering SVGD in the case of $n = 1$, we can frame the density q represented by a single particle $\boldsymbol{x}'$ as

$$q(\boldsymbol{x}) = \begin{cases} 1 & \boldsymbol{x} = \boldsymbol{x}' \\ 0 & \boldsymbol{x} \neq \boldsymbol{x}' \end{cases} \,. \quad (21)$$

The KL divergence between $q(\boldsymbol{x})$ and the target density $p(\boldsymbol{x})$ can then be simplified as

$$\text{KL}(q\|p) = \mathbb{E}_{q(\boldsymbol{x})}[\log(q(\boldsymbol{x})/p(\boldsymbol{x}))] = -\log p(\boldsymbol{x}') \,. \quad (22)$$

Finally, standard SVGD in the case of $n = 1$ obtain its optimal particle by optimizing the following problem:

$$\begin{aligned} q^* &= \arg\min_q \text{KL}(q\|p) \\ &= \arg\min_{\boldsymbol{x}'} \{-\log p(\boldsymbol{x}')\} \\ &= \arg\max_{\boldsymbol{x}'} p(\boldsymbol{x}') \,, \end{aligned} \quad (23)$$

which finally concludes the proof.

Remark. In practice, this $k(\boldsymbol{x}, \boldsymbol{x}) = c$ can be well satisfied, such as the radial basis function (RBF) kernel that we have applied in our experiments.

B EXPERIMENTAL SETTINGS

B.1 THE DARTS SEARCH SPACE

In the DARTS [Liu et al., 2019] search space, each candidate architecture consists of a stack of L cells, which can be represented as a directed acyclic graph (DAG) of N nodes denoted by $\{z_0, z_1, \ldots, z_{N-1}\}$. Among these N nodes in a cell, z_0 and z_1 denote the input nodes produced by two preceding cells, and z_N denotes the output of a cell, which is the concatenation of all intermediate nodes, i.e., from z_2 to z_{N-1}. As in the work of Liu et al. [2019], to select the best-performing architectures, we need to select their corresponding cells, including the normal and reduction cell. We refer to the DARTS paper for more details. In practice, this search space is conventionally represented as a supernet stacked by 8 cells (6 normal cells and 2 reduction cells) with initial channels of 16.

B.2 MODEL TRAINING OF SUPERNET

Following [Xu et al., 2020], we apply a partial channel connection with $K = 2$ in the model training of the supernet, which allows us to accelerate and reduce the GPU memory consumption during this model training. We split the standard training dataset of CIFAR-10 into two piles in our ensemble search: 70% randomly sampled data is used in the model training of the supernet, and the rest is used to obtain the posterior distribution of neural architectures in Sec. 3.2 and also the final selected ensembles in Sec. 3.3. To achieve not only a fair but also a sufficient model training for every candidate architecture, we apply *stochastic gradient descent* (SGD) with epoch 50, learning rate cosine scheduled from 0.1 to 0, momentum 0.9, weight decay 3×10^{-4} and batch size 128 in the model training of the supernet, where only a single candidate architecture is uniformly randomly sampled from this supernet in every training step.

B.3 POSTERIOR DISTRIBUTION

Variational posterior distribution. Following [Xie et al., 2019], the variational posterior distribution of architectures is represented as $p_{\boldsymbol{\alpha}}(\mathcal{A})$ parameterized by $\boldsymbol{\alpha}$. Specifically, within the search space demonstrated in our Appendix B.1, each intermediate nodes z_i is the output of one selected operation $o \sim p_{\boldsymbol{\alpha}_i}(o)$ using the inputs from its proceeding nodes or cells, where $\mathcal{O}$ is a predefined operation set for our search. Specifically, given $\boldsymbol{\alpha}_i = (\alpha_i^{o_1} \cdots \alpha_i^{o_{|\mathcal{O}|}})$, $p_{\boldsymbol{\alpha}_i}(o)$ can

be represented as

$$p_{\boldsymbol{\alpha}_i}(o) = \frac{\exp(\alpha_i^o/\tau)}{\sum_{o\in\mathcal{O}}\exp(\alpha_i^o/\tau)}\ , \tag{24}$$

where τ denotes the softmax temperature, which is usually set to be 1 in practice. Based on this defined probability for each intermediate node z_i, our variational posterior distribution can be framed as

$$p_{\boldsymbol{\alpha}}(A) = \prod_{i=2}^{N-2} p_{\boldsymbol{\alpha}_i}(o)\ . \tag{25}$$

More precisely, this representation is applied for single-path architecture with identical cells. We use it to ease our representation. For double-path architectures consisting of two different cells (i.e., normal and reduction cell), e.g., the candidate architecture in the DARTS search space, a similar representation can be obtained.

Optimization details. To optimize (9), we firstly relax our variational posterior distribution to be differentiable using the Straight-Through (ST) Gumbel-Softmax [Maddison et al., 2017, Jang et al., 2017] with the reparameterization trick. More precisely, we propose a variant of ST Gumbel-Softmax outputting the double-path architectures in the DARTS search space. Then, we use stochastic gradient-based algorithms to optimize (9) efficiently. In each optimization step, we sample one neural architecture from the distribution $p_{\boldsymbol{\alpha}}(\mathcal{A})$ to estimate $\mathbb{E}_{\mathcal{A}\sim p_{\boldsymbol{\alpha}}(\mathcal{A})}\left[\log p(\mathcal{D}|\mathcal{A})\right]$ (i.e., the commonly used Cross-Entropy loss). In practice, we use Adam [Kingma and Ba, 2015] with learning rate 0.01, $\beta_1 = 0.9$, $\beta_2 = 0.999$ and weight decay 3×10^{-4} to update our variational posterior distribution $p_{\boldsymbol{\alpha}}(\mathcal{A})$ for 20 epochs.

B.4 SVGD OF REGULARIZED DIVERSITY

Continuous relaxation of variational posterior distribution. Notably, SVGD [Liu and Wang, 2016] and also our SVGD-RD is applied for continuous distribution. Unfortunately, the variational posterior distribution $p_{\boldsymbol{\alpha}}(\mathcal{A})$ is discrete due to a discrete search space. To apply SVGD-RD, we firstly relax this discrete posterior into its continuous counterpart using a mixture of Gaussian distribution. Specifically, we represent each operation $o \in \mathcal{O}$ in (24) into a one-hot vector $\boldsymbol{h}_o$. By introducing the random variable $\boldsymbol{o}_i \in \mathbb{R}^{|\mathcal{O}|}$ and multi-variate normal distribution $\mathcal{N}(\boldsymbol{o}_i|\boldsymbol{h}_o, \Sigma)$ into our relaxation, our relaxed posterior distribution of neural architectures can be framed as

$$\widehat{p}_{\boldsymbol{\alpha}}(\mathcal{A}) = \prod_{i=2}^{N-2} 1/Z_i \sum_{o\in\mathcal{O}} p_{\boldsymbol{\alpha}_i}(o)\mathcal{N}(\boldsymbol{o}_i|\boldsymbol{h}_o, \Sigma)\ , \tag{26}$$

where Z_i denotes the normalization constant. Given the sampled particle $\boldsymbol{x}^* = (\cdots \boldsymbol{o}_i^* \cdots)$ in SVGD-RD, the final selected architecture can then be derived using the determination of each selected operation o_i^*, i.e.,

$$o_i^* = \arg\min_{o\in\mathcal{O}} \|\boldsymbol{o}_i^* - \boldsymbol{h}_o\|_2\ . \tag{27}$$

Optimization details. Since Liu and Wang [2016] have demonstrated that SVGD is able to handle unnormalized target distributions, the normalization constant in (26) can then be ignored in our SVGD-RD algorithm. In practice, the covariance matrix Σ in (26) is set to an identity matrix scaled by $|\mathcal{O}|$. Besides, the parameter δ is optimized as a hyper-parameter via grid search or Bayesian Optimization [Snoek et al., 2012] within the range of $[-2, 1]$ in practice. To obtain well-performing particles in our SVGD-RD algorithms efficiently, we apply SGD using the gradient provided in Sec. 3.3.2 with a *radial basis function* (RBF) kernel on randomly initialized particles for L=1000 iterations under a learning rate of 0.1 and a momentum of 0.9.

B.5 EVALUATION ON BENCHMARK DATASETS

Evaluation on CIFAR-10/100. We apply the same constructions in DARTS [Liu et al., 2019] for our final performance evaluation on CIFAR-10/100: The final selected architectures consist of 20 cells, and 18 of them are identical normal cells, with the rest being the identical reduction cell. An auxiliary tower with a weight of 4 is located at the 13-th cell of the final selected architectures. The final selected architecture is then trained using stochastic gradient descent (SGD) for 600 epochs with a learning rate cosine scheduled from 0.025 to 0, momentum 0.9, weight decay 3×10^{-4}, batch size 96 and initial channels 36. Cutout [Devries and Taylor, 2017], and a scheduled DropPath, i.e., linearly decayed from 0.2 to 0, are employed to achieve SOTA generalization performance.

Evaluation on ImageNet. Following [Liu et al., 2019], the architectures evaluated on ImageNet consist of 14 cells (12 identical normal cells and 2 identical reduction cells). To meet the requirement of evaluation under the mobile setting (less than 600M multiply-add operations), the number of initial channels for final selected architectures are conventionally set to 44. We adopt the training enhancements in Liu et al. [2019], Chen et al. [2019], Chen and Hsieh [2020], including an auxiliary tower of weight 0.4 and label smoothing. Following P-DARTS Chen et al. [2019] and SDARTS-ADV Chen and Hsieh [2020], we train the selected architectures from scratch for 250 epochs using a batch size of 1024 on 8 GPUs, SGD optimizer with a momentum of 0.9 and a weight decay of 3×10^{-5}. The learning rate applied in this training is warmed up to 0.5 for the first 5 epochs and then decreased to zero linearly.

B.6 ADVERSARIAL DEFENSE

Adversarial attack intends to find a small change for each input such that this input with its corresponding small change will be misclassified by a model. As ensemble is known to be a possible defense against such adversarial attacks [Strauss et al., 2017], we also examine the effectiveness of our NESBS algorithm by comparing the model robust-NESBS achieved by our algorithms to other ensemble and ensemble search algorithms under various benchmark adversarial attacks. To the best of our knowledge, we are the first to examine the advantages of ensemble search algorithms in defending against adversarial attacks.

In this experiment, two processes are required, i.e., *attack* and *defense*. The *attack* process is a typical white-box attack scenario: Only a single model (randomly sampled from an ensemble) is attacked by an attacker, and this process will be repeated for n rounds given an ensemble of size n in order to accurately measure the improvement of model robust-NESBS induced by an ensemble. In each round, a different model from this ensemble is selected to be attacked. The *defense* process is then applied using neural network ensembles, i.e., neural network ensembles will make predictions based on those perturbed images produced by the aforementioned attacker. Corresponding to the *attack* process, we also need to repeat this defense process for n rounds. In fact, such an adversarial defense setting is reasonably practical when only a single model from an ensemble is required to be publicly available for model producers.

We apply the following attacks in this experiment: The *Fast Gradient Signed Method* (FGSM) attack Goodfellow et al. [2015], the *Projected Gradient Descent* (PGD) attack Madry et al. [2018], and also the *Carlini Wagner* (CW) attack Carlini and Wagner [2017]. In both the FGSM attack and the PGD attack, we impose a L_∞ norm constrain of 0.01. The step size and the number of iterations in the PGD attack are set to 0.008 and 40, respectively. We adopt the same configurations of the CW attack under a L_2 norm constrain in [Carlini and Wagner, 2017]: We set the confidence constant, the range of constant c, the number of binary search steps, and the maximum number of optimization steps to 0, $[0.001, 10]$, 3, and 50, respectively; we then adopt Adam [Kingma and Ba, 2015] optimizer with learning rate 0.01 and $\beta_1 = 0.9$, $\beta_2 = 0.999$ in its search process.

C COMPLEMENTARY RESULTS

C.1 ENSEMBLE PERFORMANCE ESTIMATION

As shown in Sec. 3.1, we apply the model parameters inherited from a trained supernet to estimate the performance of candidate architectures as well as their ensembles in our NESBS algorithm. We therefore use the following three metrics to measure the effectiveness of such estimation in the DARTS search space: the Spearman's rank order coefficient between the estimated and true performances, the Pearson correlation coefficient between the estimated and true performances, and the percentage of architectures achieving both Top-k estimated performance and Top-k true performances (named the Agreement-k). Since the evaluation of the true performances is prohibitively costly, we randomly sample 10 architectures of diverse estimated performances from the DARTS search space for this experiment. Notably, based on these 10 architectures, there are hundreds of possible ensembles under the ensemble size of 3, 5, 7, which we believe is sufficiently large to validate the effectiveness of our performance estimations. To obtain the true performance of candidate architectures as well as their ensembles, we train these architectures independently for 100 epochs following the settings in Appendix B.5.

Metric	$n = 1$	$n = 3$	$n = 5$	$n = 7$
Spearman	0.65	0.33	0.40	-0.12
Pearson	0.82	0.45	0.45	-0.16
Agreement-30%	33%	20%	31%	25%

Table 6: The correlation between the estimated and true performances of candidate architectures and their ensembles in the DARTS search space on CIFAR-10.

Table 6 summarizes the results. Notably, the estimated and true performances are shown to be positively correlated in the case of n=$1, 3, 5$ by achieving relatively high Spearman and Pearson coefficients as well as a high agreement in these cases. Although the coefficients are low when the ensemble size is larger (i.e., n=7), the estimated and true performances are still capable of achieving a reasonably good agreement in this case. Based on these results, we argue that our estimated ensemble performance is informative and effective for our ensemble search. This effectiveness can also be supported by the competitive search results achieved by our NESBS in Sec. 4.2.

C.2 POST-TRAINING VS. BEST-RESPONSE POSTERIOR DISTRIBUTION

To examine the advantages of our post-training posterior distribution, we compare it with its best-response counterpart applied in [Dong and Yang, 2019a, Xie et al., 2019]. While our post-training posterior distribution is obtained *after* the model training of the supernet, the best-response posterior distribution is updated *during* the model training of the supernet. We refer to [Dong and Yang, 2019a, Xie et al., 2019] for more details about this best-response posterior distribution. We follow the optimization details in Appendix B.2 and B.3 to obtain these two posterior distributions.

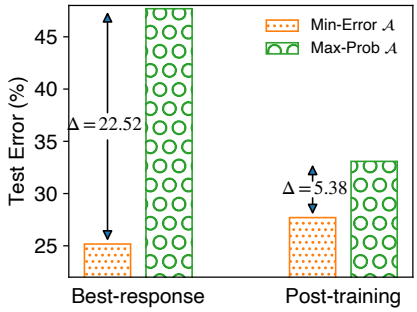


Figure 4: The comparison of performance discrepancy with the post-training and best-response posterior distribution on CIFAR-10. This performance discrepancy is measured by the gap of test error between the best-performing architecture (i.e., the architecture with the smallest test error) and the maximal-probability architecture (i.e., the architecture with the largest probability in the corresponding posterior distribution) in the DARTS search space.

More accurate characterization of single-model performances using post-training posterior distribution. We firstly compare the characterization of single-mode performance using these two posterior distributions by examining the performance discrepancy between their best-performing architecture (i.e., the architecture achieving the smallest test error) and maximal-probability architecture (i.e., the architecture achieving the largest probability in the corresponding posterior distribution) in the search space. In this experiment, the performance discrepancy is measured by the gap of test error achieved by the best-performing architecture and the maximal-probability architecture using the model parameters inherited from the supernet.

Figure 4 illustrates the comparison. The results show that our post-training posterior distribution enjoys a smaller performance discrepancy, suggesting that our post-training posterior distribution is able to provide a more accurate characterization of the single-model performances. Interestingly, the best-response counterpart contributes to the best-performing architecture with a lower test error than our post-training posterior distribution, which should result from the Matthew Effect as justified in [Hong et al., 2020]. Specifically, well-performing architectures contribute to the frequent selections of these architectures for their model training during the optimization of the best-response posterior distribution. This will finally result in unfair model training in the search space and therefore the inaccurate characterization of single-model performances. Notably, we need a more accurate characterization of single-mode performance in this paper, as shown in Sec. 3.2. Therefore, our post-training posterior distribution should be more suitable than its best-response counterpart in our ensemble search.

Method	Best-response	Post-training
NESBS (MC Sampling)	4.74	**4.54**$_{\Delta=0.20}$
NESBS (SVGD-RD)	4.81	**4.48**$_{\Delta=0.33}$

Table 7: The comparison of true ensemble test error (%) on CIFAR-10 achieved by our NESBS algorithm using the post-training posterior distribution and its best-response counterpart with an ensemble size of n=3. We use Δ to denote the improved generalization performance achieved by our post-training posterior distribution.

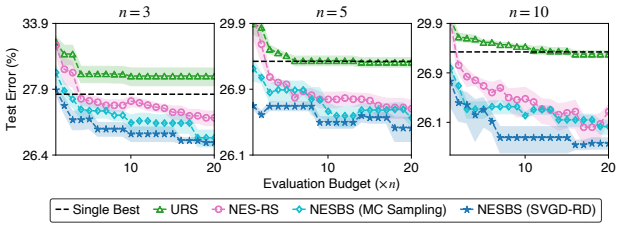


Figure 5: The comparison of search effectiveness (test error of ensembles in the y-axis) and efficiency (evaluation budget in the x-axis) for different ensemble search algorithms under varying ensemble size n. The single best baseline refers to the single best architecture achieving the lowest test error in the search space. The y-axis is shown in log-scale to ease visualization. Note that the test error for each algorithm is reported with the mean and standard error of five independent trials.

Improved performance of selected ensembles using post–training posterior distribution. We then compare the final ensemble test performance achieved by our NESBS algorithm using the post-training posterior distribution and its best-response counterpart on CIFAR-10 with the ensemble size of $n = 3$. To obtain the final ensemble performance, we train each architecture in an ensemble for 100 epochs following the settings in Appendix B.5. Table 7 summarizes the results. Notably, our post-training posterior distribution is shown to be capable of contributing to an improved ensemble performance than its best-response counterpart, which further demonstrates the advantages of applying the post-training posterior distribution in our ensemble search.

C.3 EFFECTIVENESS AND EFFICIENCY

As justified in Sec 3.3, both our MC Sampling and SVGD-RD algorithms can sample neural architectures with competitive single-model performances and diverse model predictions, which are known to be the criteria for well-performing ensembles [Zhou, 2012]. To further demonstrate that our algorithms are capable of selecting well-performing ensembles effectively and efficiently based on this sampling property, we compare our NESBS algorithm, including NESBS (MC Sampling) and NESBS (SVGD-RD), with the follow-

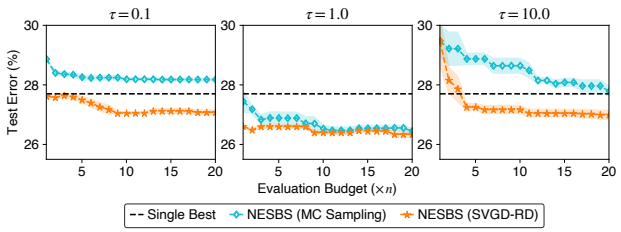


Figure 6: The comparison of search effectiveness (test error of ensembles in the y-axis) and efficiency (evaluation budget in the x-axis) between our NESBS (MC Sampling) and NESBS (SVGD-RD) algorithm under varying softmax temperature τ. The single best baseline refers to the single best architecture achieving the lowest test error in the search space. Each test error is reported with the mean and standard error of five independent trials.

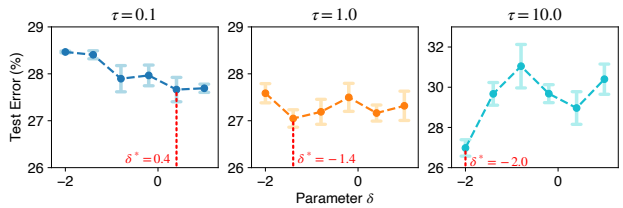


Figure 7: The comparison of ensemble test error achieved by our NESBS (SVGD-RD) algorithm with varying δ under different softmax temperature τ given an ensemble size of $n = 5$. We use δ^* to denote the optimal δ we obtained in our SVGD-RD algorithm under different temperature τ. The test error for each δ is reported with the mean and standard error of five independent trials.

ing ensemble search baselines on CIFAR-10 [Krizhevsky et al., 2009] in the DARTS [Liu et al., 2019] search space: (a) Uniform random sampling which we refer to as URS, and (b) NES-RS [Zaidi et al., 2021]. That is, we only replace the Bayesian sampling in our NESBS algorithm with these two different sampling/selection algorithms in this experiment and we keep using the model parameters inherited from a supernet to estimate the single-model and ensemble performances of architectures (including the test errors). The detailed experimental settings are in Appendix B.

Figure 5 illustrates the search results. Note that both NES-RS and our NESBS are able to achieve lower test errors than the single best-performing architecture in the search space. These results therefore demonstrate that these two ensemble search algorithms are indeed capable of achieving improved performance over conventional NAS algorithms that select only one single architecture from the search space. More importantly, given the same evaluation budgets, our NESBS algorithm consistently achieves lower test errors than URS and NES-RS, indicating the superior search effectiveness achieved by our NESBS algorithm. Meanwhile, our NESBS algorithm requires fewer evaluation budgets than URS and NES-RS to achieve comparable test errors, which also suggests that our algorithm is more efficient than URS and NES-RS. Interestingly, compared with MC Sampling, SVGD-RD can consistently produce improved search effectiveness and efficiency, which likely results from its controllable trade-off between the single-model performances and the diverse model predictions as justified in Sec. 3.3. Overall, these results have well justified the effectiveness and efficiency of our NESBS algorithm.

C.4 THE ADVANTAGES OF CONTROLLABLE DIVERSITY IN SVGD-RD

To examine the advantages of controllable diversity in our SVGD-RD, we firstly compare the search effectiveness and efficiency achieved by our NESBS (MC Sampling) and NESBS (SVGD-RD) algorithm with varying softmax temperature τ (appeared in (24)). A larger temperature τ will lead to a flatter posterior distribution and hence degenerate its capability of characterizing single-model performances of neural architectures as indicated in (24). We use these posterior distributions with varying temperature τ to simulate the possible posterior distributions we may obtain in practice. Figure 6 illustrates the comparison on CIFAR-10 in the DARTS search space with an ensemble size of $n = 5$. Notably, our NESBS (SVGD-RD) with controllable diversity can consistently achieve improved search effectiveness and efficiency than our NESBS (MC Sampling). Interestingly, this improvement becomes larger in the case of $\tau = 0.1, 10.0$, which should be the consequences of a bad exploration and exploitation achieved by our NESBS (MC Sampling), respectively. These results therefore suggest that the controllable diversity in our SVGD-RD generally can lead to improved search effectiveness and efficiency than our NESBS (MC Sampling).

We further provide the comparison of ensemble test error achieved by our SVGD-RD with varying δ under different softmax temperature τ in Figure 7. Notably, when the posterior distribution tends to be flatter (i.e., $\tau = 10$), a smaller δ is preferred by our SVGD-RD in order to sample architectures with better single-model performances while maintaining the compelling diverse model predictions. Meanwhile, when this posterior distribution tends to be sharper (i.e., $\tau = 0.1$), a larger δ is preferred by our SVGD-RD in order to sample architectures with a larger diverse model predictions while preserving the competitive single-model performances. Based on this controllable diversity and hence the controllable trade-off between the single-model performances and the diverse model predictions, our SVGD-RD is thus capable of achieving comparable performances under varying τ, which usually improve over our NESBS (MC Sampling) by comparing them with the results in Figure 6. These results further validate the advantages of the controllable diversity in our SVGD-RD.