Going Beyond Neural Architecture Search with Sampling-based Neural Ensemble Search

Yao Shu, Yizhou Chen, Zhongxiang Dai, Bryan Kian Hsiang Low

1 Dept. of Computer Science, National University of Singapore, Republic of Singapore
{shuyao, ychen041, daizhongxiang, lowkh}@comp.nus.edu.sg

Abstract

Recently, Neural Architecture Search (NAS) has been widely applied to automate the design of deep neural networks. Various NAS algorithms have been proposed to reduce the search cost and improve the generalization performance of those final selected architectures. However, these NAS algorithms aim to select only a single neural architecture from the search spaces and thus have overlooked the capability of other candidate architectures in helping improve the performance of their final selected architecture. To this end, we present two novel sampling algorithms under our Neural Ensemble Search via Sampling (NESS) framework that can effectively and efficiently select a well-performing ensemble of neural architectures from NAS search space. Compared with state-of-the-art NAS algorithms and other well-known ensemble search baselines, our NESS algorithms are shown to be able to achieve improved performance in both classification and adversarial defense tasks on various benchmark datasets while incurring a comparable search cost to these NAS algorithms.

1 Introduction

Recent years have witnessed a surging interest in designing well-performing neural 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 neural 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 neural architectures. Since then, various algorithms (Pham et al. 2018; Chen et al. 2019; Liu, Simonyan, and Yang 2019; Chen and Hsieh 2020) have been developed to improve the search efficiency (i.e., the search cost) or the search effectiveness (i.e., the performance of their final selected architectures) of 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 in helping improve the performance of their final selected architecture. Meanwhile, neural network ensembles are widely known to be capable of achieving an improved performance compared with a single neural network on various tasks (Gal and Ghahramani 2016; Cortes et al. 2017; Lakshminarayanan, Pritzel, and Blundell 2017). This naturally leads to the question for NAS: How to select an ensemble of architectures from the NAS search space to achieve improved performances than conventional NAS algorithms?

To the best of our knowledge, only limited efforts (Zaidi et al. 2020) have been devoted to this topic in the NAS literature. Specifically, the Neural Ensemble Search (NES) algorithms proposed by (Zaidi et al. 2020), which are based on random search or evolutionary algorithm, can only achieve limited ensemble performances while requiring excessive search costs.

To this end, this paper introduces two novel sampling algorithms under our Neural Ensemble Search via Sampling (NESS) framework, which can select well-performing ensembles of architectures effectively and efficiently from the NAS search space. Specifically, we firstly represent the NAS search space as a supernet following DARTS (Liu, Simonyan, and Yang 2019) and then use the model parameters inherited from this trained supernet to estimate the ensemble performance of independently trained neural architectures (Sec. 3.1). Next, since both the single-model performances and the predictive diversity contribute to the final ensemble performance (Zhou 2012), we derive a variational posterior distribution of neural architectures, which is able to provide a good characterization for these two essential factors (Sec. 3.2). This finally results in two efficient sampling algorithms under our NESS framework, i.e., Monte Carlo sampling (MC Sampling) and Stein Variational Gradient Descent with regularized diversity (SVGD-RD), that can effectively and efficiently select ensembles of neural architectures satisfying the criteria for well-performing ensembles (Zhou 2012), i.e., with both competitive single-model performances and compelling predictive diversity (Sec. 3.3).

We empirically demonstrate the search effectiveness and efficiency of our NESS algorithms in the DARTS search space on CIFAR-10 (Sec. 4.1). The results show that our algorithms can achieve improved search effectiveness and efficiency than other ensemble search baselines. Moreover, our final selected ensembles are shown to outperform state-
of-the-art NAS, ensemble, and ensemble search algorithms in both classification and adversarial defense tasks on various benchmark datasets while incurring a comparable search cost to these NAS algorithms (Sec. 4.2). Finally, we justify that the improved performances achieved by our NESS algorithms result from their capability of achieving a favorable trade-off between competitive single-model performances and compelling predictive diversity (Sec. 4.3).

2 Related Works and Background

2.1 Neural Architecture Search

Various algorithms (Zoph and Le 2017; Zoph et al. 2018; Real et al. 2019) have been proposed to automate the design of neural architectures in the literature. However, these NAS algorithms are inefficient due to their requirement of independent model training for each candidate architecture in the search space. To reduce such training costs, a supernet has been introduced by Pham et al. (2018) to share model parameters among candidate architectures in the search space. Consequently, only the model training of this supernet is required. Following the work of Pham et al. (2018), numerous NAS algorithms based on parameter sharing (Chen et al. 2019; Liu, Simonyan, and Yang 2019; Xie et al. 2019; Chen and Hsieh 2020; Chu et al. 2020) have been developed. Nonetheless, these algorithms aim to select only a single architecture with the best performance from the search space.

2.2 Neural Network Ensembles

Neural network ensembles have been widely applied to improve the performance of a single neural network on different tasks (Dieterich 2000). Over the years, various methods have been proposed to construct such neural network ensembles. For example, Gal and Ghahramani (2016) have proposed to use Monte Carlo Dropout to obtain neural network ensembles at test time. Meanwhile, deep ensembles (DeepEns) (Lakshminarayanan, Pritzel, and Blundell 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 ensembles (Huang et al. 2017a).

More recently, Zaidi et al. (2020) have proposed Neural Ensemble Search (NESS) algorithms to select diverse neural architectures from the NAS search space to build ensembles. However, their algorithm results in unaffordable search costs and limited ensemble performance. Instead, our NESS algorithms in this paper advance this line of works (i.e., NES) by achieving state-of-the-art (SOTA) performances for ensembles of neural architectures while incurring a comparable search cost to conventional NAS algorithms.

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 \( p(x) \) with a simpler density \( q^*(x) \) in a predefined set \( \mathcal{Q} \), which is selected by minimizing the Kullback-Leibler (KL) divergence between these two densities:

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

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

\[
x_i \leftarrow x_i + \epsilon \hat{\phi}^*(x_i), \quad \forall i \in \{1, \cdots, n\}. \tag{2}
\]

Let \( q_{\epsilon[\phi]} \) denote the distribution of updated particles \( x' = x + \epsilon \phi(x) \). 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 \( k(x, x') \). 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 \( \phi^* \) in (2) can be obtained by solving the following problem:

\[
\phi^* = \mathop{\arg\max}_{\phi \epsilon \mathbb{F}} \left\{ -\frac{d}{d\epsilon} KL(q_{\epsilon[\phi]}||p)|_{\epsilon=0} \right\}, \tag{3}
\]

which yields a closed-form solution:

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

In practice, Liu and Wang (2016) have approximated the expectation in this close-form solution with the empirical mean of particles: \( \tilde{\phi}^*(x_i) \approx \hat{\phi}^*(x_i) \), where \( \hat{\phi}^*(x_i) \) is defined as

\[
\tilde{\phi}^*(x_i) \triangleq \frac{1}{n} \sum_{j=1}^n k(x_j, x_i)\nabla_{x_i} \log p(x_j) + \nabla_{x_i} k(x_j, x_i). \tag{5}
\]

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

3 Sampling-based Neural Ensemble Search

Let \( f_A(x, \theta_A) \) denote the output of an architecture \( A \) with input data \( x \) and model parameter \( \theta_A \). Let \( S \) and \( \Theta_S \) denote a set of architectures and their corresponding model parameters, respectively. Given the ensemble scheme \( f_S(x, \Theta_S) \triangleq 1/n \sum_{A \in S} f_A(x, \theta_A) \) with an ensemble size of \( |S| = n \), let \( \mathcal{L}_{\text{train}} \) and \( \mathcal{L}_{\text{val}} \) denote the training and validation loss respectively. Neural Ensemble Search (NESS) (Zaidi et al. 2020) can then be framed as a bi-level combinatorial optimization problem:

\[
\min_{S} \mathcal{L}_{\text{val}}(f_S(x, \Theta_S^*)) \quad \text{s.t.} \quad \forall \theta_A^* \in \Theta_S^*, \quad \theta_A^* = \mathop{\arg\min}_{\theta_A} \mathcal{L}_{\text{train}}(f_A(x, \theta_A)). \tag{6}
\]

Note that (6) is challenging to solve due to the following reasons: (I) The enormous number of candidate architectures in the NAS search space, e.g., \( \sim 10^{25} \) in the DARTS

\footnote{We apply such an ensemble scheme for simplicity following the work of (Zaidi et al. 2020).}
search space (Liu, Simonyan, and Yang 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 \( \sim n^{m} \) 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. (2020) have attempted to avoid these two problems by selecting a small fraction of candidate architectures from the original search space for their final ensemble search. Nonetheless, they fail to explore the whole search space and therefore may only achieve limited ensemble performances. Furthermore, their search cost is unaffordable due to the independent model training of every selected architecture.

To solve (6) both effectively and efficiently, we present two novel sampling algorithms under our Neural Ensemble Search via Sampling (NESS) framework. Particularly, we firstly employ the model parameters inherited from a supernet to estimate the ensemble performance of neural architectures (Sec. 3.1), which only requires the model training of this supernet and thus allows us to overcome the aforementioned challenge (I). We then derive a posterior distribution of neural architectures to characterize both the single-model performances and the predictive diversity of candidate architectures in the search space (Sec. 3.2). Finally, given this posterior distribution, we propose to apply Monte Carlo Sampling (MC Sampling) and Stein Variational Gradient Descent with regularized diversity (SVGD-RD) under our NESS framework to select a set of neural architectures with both compelling single-model performances and diverse model predictions required by well-performing ensembles (Zhou 2012) (Sec. 3.3). This allows us to overcome the aforementioned challenge (II).

### 3.1 Ensemble Performance Estimation

Following one-shot NAS algorithms (Pham et al. 2018; Liu, Simonyan, and Yang 2019), we represent the entire NAS search space as a supernet. This finally allows us to use the model parameters inherited from this trained supernet to estimate the performances of candidate architectures as well as their ensembles in the search space. To achieve accurate estimation of these performances, we need to ensure training fairness such that every candidate architecture in the search space is trained for a comparable number of steps (Chu et al. 2019). Specifically, following (Chu et al. 2019; Guo et al. 2020), in each training step of this supernet, we uniformly randomly sample a single candidate architecture from this supernet for the model training (more details in Appendix B.2). The training fairness of such a training scheme can then be guaranteed, as demonstrated in Appendix A. Moreover, Appendix C.1 empirically validates the effectiveness of our performance estimations.

### 3.2 Posterior Distribution of Neural Architectures

It has been demonstrated that both competitive single-model performances and diverse model predictions are required to achieve compelling ensemble performances (Zhou 2012). Fortunately, these two factors can be characterized by a posterior distribution of neural architectures. Specifically, let \( D \) denote the validation dataset, and let \( p(A) \) and \( p(A|D) \) denote the prior and posterior distributions of candidate architectures in the search space, where \( p(A) \) follows from a categorical uniform distribution as required in Sec. 3.1. According to the Bayes’ theorem, given this uniform prior distribution \( p(A) \) and constant \( p(D) \), we have that

\[
p(A|D) = \frac{p(D|A)p(A)}{p(D)} = \frac{p(D|A)}{p(D)} \propto p(D|A) .
\]

Note that \( p(D|A) \) (i.e., the likelihood) can exactly be used to represent the single-model performances of an architecture \( A \) in practice. Therefore, (7) implies that the posterior distribution \( p(A|D) \) can also be used to characterize the single-model performances of architectures in the search space.

Suppose the loss function \( L(f) \) is \( \gamma \)-Lipschitz continuous, and let \( \| f_{A_1} - f_{A_2} \|_2 \) denote the predictive diversity of architectures \( A_1 \) and \( A_2 \). We can then approximate this diversity with its lower bound induced by the Lipschitz continuity of the loss function:

\[
\| f_{A_1} - f_{A_2} \|_2 \geq \gamma^{-1} \| L(f_{A_1}) - L(f_{A_2}) \|_2 .
\]

Interestingly, (8) suggests that in addition to being able to characterize the single-model performances of neural architectures (i.e., \( L(f) \)) as shown above, the posterior distribution \( p(A|D) \) can also be employed to estimate the predictive diversity of different architectures. Specifically, we can use \( \| p(A_1|D) - p(A_2|D) \|_2 \) to approximate the predictive diversity of architectures \( A_1 \) and \( A_2 \).

However, it is intractable to obtain exact posterior distribution \( p(A|D) \) in the NAS search space. So, we approximate it with a variational distribution \( p_\alpha(A) \) (parameterized by a low-dimensional \( \alpha \)) that can be optimized via variational inference, i.e., by minimizing the KL divergence between \( p_\alpha(A) \) and \( p(A|D) \). 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 \( p_\alpha(A) \):

\[
\alpha^* = \arg \max_\alpha \mathbb{E}_{A \sim p_\alpha(A)} [\log p(D|A)] - KL[p_\alpha(A)||p(A)] ,
\]

Similar to (Kingma and Welling 2014), stochastic gradient-based algorithms with the reparameterization trick (see Appendix B.3) can be employed to solve (9) efficiently. Interestingly, Pham et al. (2018); Xie et al. (2019) have adopted a similar form to (9) (without the KL term) during the model training of the supernet, which is called best-response posterior distribution by us. Interestingly, our post-training posterior distribution is shown to be able to provide a more accurate characterization of single-model performances and can result in an improved ensemble performance after the neural ensemble search, as we demonstrate in Appendix C.2.

### 3.3 Neural Ensemble Search via Sampling

After obtaining the posterior distribution of neural architectures (Sec. 3.2), we employ our Neural Ensemble Search via Sampling (NESS) framework to solve (6) effectively and efficiently. Specifically, we iteratively sample \( T \) ensembles
Algorithm 1: Neural Ensemble Search via Sampling (NESS)

1: **Input:** Iterations $T$, ensemble size $n$, model parameters $\theta^*$ of supernet, posterior $p_{\alpha^*}(A)$
2: for iteration $t = 1, \ldots, T$ do
3:   Sample $S_t$ of size $n$ via Algorithm 2 or 3
4:   Evaluate estimated $L_{\text{val}}(F_{S_t}(x, \Theta_{S_t}^*))$ given $\theta^*$
5: end for
6: Select the optimum $S^* = \arg\min_{S_t} L_{\text{val}}(F_{S_t}(x, \Theta_{S_t}^*))$

Algorithm 2: Monte Carlo Sampling (MC Sampling)

1: **Input:** Ensemble size $n$, set $S = \emptyset$, posterior $p_{\alpha^*}(A)$
2: for iteration $i = 1, \ldots, n$ do
3:   Sample $A_i \sim p_{\alpha^*}(A)$
4:   $S \leftarrow S \cup \{A_i\}$
5: end for
6: **Output:** $S$

Algorithm 3: SVGD with Regularized Diversity (SVGD-RD)

1: **Input:** Diversity coefficient $\delta$, ensemble size $n$, iterations $L$, initial particles $\{x_i^{(0)}\}_{i=1}^n$, posterior $p_{\alpha^*}(A)$, kernel $k(x, x')$, step size $\{\epsilon_i\}_{i=1}^L$
2: for iteration $t = 0, \ldots, L-1$ do
3:   $\phi_i^*(x) = \frac{1}{n} \sum_{j=1}^n \nabla_{x_i^{(t)}} k(x_j^{(t)}, x) - \delta \nabla_{x_i^{(t)}} k(x_i^{(t)}, x) + k(x_i^{(t)}, x) \log p_{\alpha^*}^*$
4:   $x_i^{(t+1)} \leftarrow x_i^{(t)} + \epsilon_i \phi_i^*(x_i^{(t)})$
5: end for
6: **Output:** $S = \{A_i\}_{i=1}^n$ derived based on $\{x_i^{(L)}\}_{i=1}^n$

from this posterior distribution and then select the one with a smallest estimated ensemble error, i.e., $L_{\text{val}}(F_S(x, \Theta_*^*))$, in our NESS framework (see Algorithm 1). Based on this NESS framework, we then present two novel sampling algorithms (MC Sampling and SVGD-RD) that can effectively and efficiently sample well-performing ensembles of neural architectures, i.e., ensembles in which the architectures achieve both compelling single-model performances and diverse model predictions (Zhou 2012).

Monte Carlo Sampling (MC Sampling) Given the posterior distribution of neural architectures obtained in Sec. 3.2, we firstly propose to employ Monte Carlo sampling (MC Sampling) to sample a set of neural architectures from this posterior distribution (Algorithm 2). Specifically, MC Sampling guarantees that neural architectures with better single-model performances will be sampled (i.e., exploited) with higher probabilities. Meanwhile, 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 only select a single best-performing architecture in the search space (Dong and Yang 2019; Xie et al. 2019), our MC sampling algorithm further extends these algorithms by exploring the capability of diverse neural architectures while preserving its exploitation of neural architectures with compelling single-model performances.

SVGD with Regularized Diversity (SVGD-RD) Unfortunately, the diversity of sampled architectures using our MC Sampling algorithm above is uncontrollable. 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 usually cannot represent the target distribution well, e.g., lack of diversity among those sampled particles as observed in (Zhuo et al. 2018). Moreover, this diversity of sampled particles in SVGD is still uncontrollable by human experts.

We hence propose our SVGD with regularized diversity (SVGD-RD) algorithm, which is amenable to achieving 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 $\{x_i^*\}_{i=1}^n$, we modify the objective in (1) by adding a term representing the (controllable) diversity among the particles measured by the kernel function $k(x, x')$:

$$q^* = \arg\min_{q \in \mathcal{Q}} KL(q || p) + n \delta E_{x, x' \sim q} [k(x, x')]$$

where $\delta$ is the parameter explicitly controlling the diversity. Here, $p$ in (10) denotes the posterior distribution $p_{\alpha^*}(A)$ derived in Sec. 3.2 which we intend to sample from. Following
the work of SVGD, \( q^* \) in (10) is represented by the particles \( \{ x^*_i \}_{i=1}^n \), denoting our final selected ensemble of architectures that can achieve both competitive single-model performances (i.e., large probability density) and diverse model predictions. Our Proposition 1 below provides one possible update rule for the particles \( \{ x_i \}_{i=1}^n \) 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 the details on how to optimize (10) in practice. After obtaining those optimal particles \( \{ x^*_i \}_{i=1}^n \) in our SVGD-RD algorithm, we can apply these particles to derive the neural architectures in our final selected ensembles (see details in Appendix B.4).

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

\[
 x_i \leftarrow x_i + 1/n \sum_{j=1}^n k(x_j, x_i) \nabla x_j \log p(x_j) + \nabla x_i k(x_j, x_i) - \delta \nabla x_i k(x_j, x_i)
\]

Interestingly, compared with MC Sampling, our SVGD-RD algorithm provides a controllable trade-off between the single-model performances and the predictive diversity. On the one hand, the minimization of the KL divergence in (10) encourages the selection of neural architectures with competitive performances by favoring particles with high probability densities, as shown by Proposition 2 below (proof given in Appendix A).³ On the other hand, the maximization of the scaled distance (measured by \(-n\delta E_{x, x'} k(x, x')\)) among the sampled particles leads to a controllable diversity (via \(\delta\)) among these sampled particles (see Figure 1).

**Proposition 2.** Let \( p \) denote the target density and \( k(x, x') \) be a constant when \( x = x' \). For any \( \delta \in \mathbb{R} \), our SVGD-RD algorithm in the case of \( n = 1 \) will be equivalent to the maximization of the target density \( p \).

4 Experiments

4.1 Search 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 demonstrate that our algorithms are capable of selecting well-performing ensembles effectively and efficiently based on this sampling property, we compare our NESS algorithms, i.e., NESS (MC Sampling) and NESS (SVGD-RD), with the following ensemble search baselines on CIFAR-10 (Krizhevsky, Hinton et al. 2009) in the DARTS (Liu, Simonyan, and Yang 2019) search space: (a) Uniform random sampling under the NESS framework which we refer to as NESS (RS), and (b) NES-RS (Zaidi et al. 2020). The detailed experimental settings are shown in Appendix B.

³Although Proposition 2 can only be applied 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 1.

Figure 2 shows the search results. Notably, both our NESS algorithms and NES-RS are able to achieve lower test errors than the single best architecture in the search space. This implies the improved effectiveness of these two ensemble search algorithms over conventional NAS algorithms that only selects a single architecture from the search space. In addition, given the same evaluation budgets, our NESS algorithms consistently achieve lower test errors than NESS (RS) and NES-RS, which indicates the superior search effectiveness of our NESS algorithms. Meanwhile, our NESS algorithms require fewer evaluation budgets than NESS (RS) and NES-RS to achieve comparable test errors, which suggests that our algorithms are also more efficient than NESS (RS) and NES-RS. Interestingly, compared with MC Sampling, our 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 predictive diversity as justified in Sec. 3.3. Overall, these results have justified the effectiveness and efficiency of our NESS algorithms.

4.2 Evaluation of Selected Ensembles

We further demonstrate the superior search effectiveness and efficiency of our NESS algorithms by comparing their final selected ensembles of architectures with well-known NAS, ensemble, and ensemble baselines in various tasks on CIFAR-10/100. We follow Appendix B.5 to evaluate the final ensembles selected by our NESS algorithms with \( n = 3 \) and \( T = 5 \) in Sec. 4.1. Additional evaluations on ImageNet (Deng et al. 2009) are provided in Appendix C.3.

**Ensemble for classification.** Table 1 summarizes the comparison of classification performances. Compared with popular NAS algorithms, our NESS algorithms are able to achieve improved generalization performances while incurring a comparable search cost. Even compared with other ensemble methods such as MC DropPath (i.e., developed following Monte Carlo Dropout (Gal and Ghahramani 2016)) and deep ensembles (DeepEns) (Lakshminarayanan, Pritzel,
| Architecture(s)                        | Test Error (%) | Params (M) | Search Cost (GPU Days) | Search Method |
|---------------------------------------|----------------|------------|------------------------|---------------|
|                                       | C10            | C100       | C10                    | C100          |               |
| DenseNet-BC (Huang et al. 2017b)      | 3.46           | 17.18      | 25.6                   | 25.6          | manual        |
| NASNet-A (Zoph et al. 2018)           | 2.65           | -          | 3.3                    | -             | manual        |
| AmoebaNet-A (Real et al. 2019)        | 3.34           | 18.93      | 3.2                    | 3.1           | evolution     |
| PNAS (Liu et al. 2018)                | 3.41           | 19.53*     | 3.2                    | 3.2           | SMBO          |
| ENAS (Pham et al. 2018)               | 2.89           | 19.43*     | 4.6                    | 4.6           | RL            |
| DARTS (Liu, Simonyan, and Yang 2019)  | 2.76           | 17.54†     | 3.3                    | 3.4           | gradient      |
| GDAS (Dong and Yang 2019)             | 2.93           | 18.38      | 3.4                    | 3.4           | gradient      |
| P-DARTS (Chen et al. 2019)            | 2.50           | -          | 3.4                    | -             | gradient      |
| DARTS- (avg) (Chu et al. 2020)        | 2.59           | 17.51      | 3.5                    | 3.3           | gradient      |
| SDARTS-ADV (Chen and Hsieh 2020)      | 2.61           | -          | 3.3                    | -             | gradient      |
| 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. 2020)           | 2.50           | 15.24      | 3.0†                   | 3.1†          | 0.7           | greedy        |
| NESS (RS)                             | 2.66           | 15.95      | 3.2†                   | 3.3†          | 0.2           | sampling      |
| NESS (MC Sampling)                    | 2.41           | 14.70      | 4.3‡                   | 4.4‡          | 0.2           | sampling      |
| NESS (SVGD-RD)                        | 2.36           | 14.55      | 4.2‡                   | 4.3‡          | 0.2           | sampling      |

† Reported by Dong and Yang (2019) with their experimental settings.
‡ Obtained as the average parameter size from the architectures in an ensemble.
*
Obtained by training the corresponding architectures without cutout (Devries and Taylor 2017) augmentation.

Table 1: The comparison of generalization performances with state-of-the-art (SOTA) image classifiers on CIFAR-10/100. The results of all ensemble methods and ensemble search algorithms are reported with an ensemble size of $n = 3$. The results for MC DropPath are obtained with a drop probability of 0.01. Our search costs are evaluated on one Nvidia 2080Ti GPU.

and Blundell 2017), our algorithms can still achieve better performances in general. Note that these ensemble algorithms are orthogonal to our NESS algorithms and can thus be incorporated into our NESS algorithms for further performance improvement in practice. Furthermore, our algorithms outperform NESS (RS) and NES-RS, by achieving both improved search effectiveness (lowest test errors) and efficiency (lowest search costs). Consistent results on ImageNet can be found in Appendix C.3. Overall, these search results have further justified the effectiveness and efficiency of our NESS algorithms.

**Ensemble for adversarial defense.** Ensemble methods have been shown to be an effective defense mechanism against adversarial attacks (Strauss et al. 2017). Particularly, an adversarial attacker can only use a single model randomly sampled from an ensemble to generate the adversarial examples, and then the ensemble method makes its predictions using all models in this ensemble. Ensemble methods can usually defend against adversarial attacks in this setting because the generated adversarial examples using a single model are unlikely to fool all models in an ensemble. More details on our experimental setting are presented in Appendix B.6. Our results in Table 2 show that under different white-box adversarial attacks, our NESS algorithms can consistently achieve improved performances (i.e., higher test accuracy in the Defense columns) than other ensemble and ensemble search algorithms such as DeepEns and NES-RS. These results therefore further support the effectiveness of our NESS algorithms. Interestingly, in terms of the robustness of single models against adversarial attacks, the architectures in our final selected ensembles are also more robust (i.e., higher test accuracy in the Attack columns) than well-known NAS architectures such as DARTS and ENAS. Overall, our NESS algorithms have been shown to outperform other popular NAS, ensemble, and ensemble search algorithms by selecting architectures and ensembles with improved robustness.

### 4.3 Single-model Performances and Predictive Diversity

We demonstrate that the efficiency of our NESS algorithms results from their ability to achieve a nice trade-off between the single-model performances and the predictive diversity. We firstly quantitatively compare the single-model performances (measured by the average test error (ATE) of the models in an ensemble) and predictive diversity (measured by the pairwise predictive disagreement (PPD) of an ensemble (Fort, Hu, and Lakshminarayanan 2019)) achieved by our NESS algorithms with other ensemble and ensemble search baselines on CIFAR-10/100. We then qualitatively vi-
This paper presents two novel ensemble search algorithms, i.e., NESS (MC sampling) and NESS (SVGD-RD), that can effectively and efficiently select well-performing ensembles of architectures from the NAS search space. Empirical results show that in both classification and adversarial defense tasks on various benchmark datasets, our NESS algorithms are able to achieve improved performances compared with conventional NAS algorithms while preserving a comparable search cost. Furthermore, even compared with the ensemble search baselines, our algorithms can also enjoy improved search effectiveness and efficiency. As our algorithms rely on the effectiveness of our ensemble performance estimation (Sec. 3.1), an interesting future direction is to study how to further improve this performance estimation in order to achieve a higher ensemble performance.

### Table 2: The comparison of adversarial defense among various ensemble and ensemble search algorithms on CIFAR-10/100 under different benchmark white-box adversarial attacks.

| Dataset | Method                      | FGSM | PGD-40 | CW      |
|---------|-----------------------------|------|--------|---------|
|         | Attack (%)      | Defense (%) | Attack (%) | Defense (%) | Attack (%) | Defense (%) |
| C10     | MC DropPath (DARTS) | 2.71 ±0.39 | 16.68 ±2.63 | 2.69 ±0.30 | 70.15 ±0.29 | 0.33 ±0.04 | 8.73 ±0.43 | 0.14 ±0.05 | 26.90 ±1.37 | 0.23 ±0.07 | 69.88 ±0.16 |
|         | DeepEns (DARTS)   | 2.69 ±0.30 | 16.18 ±12.45 | 2.87 ±0.30 | 77.85 ±0.15 | 1.45 ±0.05 | 62.33 ±0.27 | 0.87 ±0.05 | 35.02 ±0.37 | 0.36 ±0.07 | 69.12 ±0.16 |
|         | NES-RS           | 2.87 ±0.30 | 17.20 ±14.14 | 3.01 ±0.30 | 69.88 ±0.16 | 1.27 ±0.05 | 65.59 ±0.27 | 0.58 ±0.05 | 35.02 ±0.37 | 0.36 ±0.07 | 69.12 ±0.16 |
|         | NESS (MC Sampling) | 2.80 ±0.30 | 16.70 ±13.84 | 2.80 ±0.30 | 69.88 ±0.16 | 1.27 ±0.05 | 65.59 ±0.27 | 0.58 ±0.05 | 35.02 ±0.37 | 0.36 ±0.07 | 69.12 ±0.16 |
|         | NESS (SVGD-RD)    | 2.78 ±0.30 | 16.50 ±13.16 | 2.78 ±0.30 | 69.88 ±0.16 | 1.27 ±0.05 | 65.59 ±0.27 | 0.58 ±0.05 | 35.02 ±0.37 | 0.36 ±0.07 | 69.12 ±0.16 |

### Table 3: The quantitative comparison of the single-model performances (measured by ATE (%), smaller is better) and the predictive diversity (measured by PPD (%), larger is better) achieved by various ensemble and ensemble search algorithms with an ensemble size of $n = 3$.

| Method                  | C10         | C100         |
|-------------------------|-------------|--------------|
|                         | ATE  | PPD       | ATE  | PPD       |
| MC DropPath (DARTS)     | 2.71 | 0.39      | 16.68 | 2.63          |
| DeepEns (DARTS)         | 2.69 | 0.30      | 16.18 | 12.45         |
| NES-RS                  | 2.87 | 0.29      | 17.20 | 14.14         |
| NESS (RS)               | 3.01 | 0.27      | 17.69 | 14.31         |
| NESS (MC Sampling)      | 2.80 | 0.27      | 16.70 | 13.84         |
| NESS (SVGD-RD)          | 2.78 | 0.27      | 16.50 | 13.16         |

The results show that in both classification and adversarial defense tasks on various benchmark datasets, our NESS algorithms are able to achieve improved performances compared with conventional NAS algorithms while preserving a comparable search cost. Furthermore, even compared with the ensemble search baselines, our algorithms can also enjoy improved search effectiveness and efficiency. As our algorithms rely on the effectiveness of our ensemble performance estimation (Sec. 3.1), an interesting future direction is to study how to further improve this performance estimation in order to achieve a higher ensemble performance.

### 5 Conclusion

This paper presents two novel ensemble search algorithms, i.e., NESS (MC sampling) and NESS (SVGD-RD), that can effectively and efficiently select well-performing ensembles of architectures from the NAS search space. Empirical results show that in both classification and adversarial defense tasks on various benchmark datasets, our NESS algorithms are able to achieve improved performances compared with conventional NAS algorithms while preserving a comparable search cost. Furthermore, even compared with the ensemble search baselines, our algorithms can also enjoy improved search effectiveness and efficiency. As our algorithms rely on the effectiveness of our ensemble performance estimation (Sec. 3.1), an interesting future direction is to study how to further improve this performance estimation in order to achieve a higher ensemble performance.
References

Carlini, N.; and Wagner, D. 2017. Towards evaluating the robustness of neural networks. In Proc. IEEE S&P, 39–57.

Chen, X.; and Hsieh, C. 2020. Stabilizing Differentiable Architecture Search via Perturbation-based Regularization. In Proc. ICML, 1554–1565.

Chen, X.; Xie, L.; Wu, J.; and Tian, Q. 2019. Progressive Differentiable Architecture Search: Bridging the Depth Gap Between Search and Evaluation. In Proc. ICCV, 1294–1303.

Chu, X.; Wang, X.; Zhang, B.; Lu, S.; Wei, X.; and Yan, J. 2020. DARTS: Robustly Stepping out of Performance Collapse Without Indicators. arXiv:2009.01027.

Chu, X.; Zhang, B.; Xu, R.; and Li, J. 2019. Fairnas: Re-thinking evaluation fairness of weight sharing neural architecture search. arXiv:1907.01845.

Cortes, C.; Gonzalvo, X.; Kuznetsov, V.; Mohri, M.; and Yang, S. 2017. AdaNet: Adaptive Structural Learning of Artificial Neural Networks. In Proc. ICML, 874–883.

Deng, J.; Dong, W.; Socher, R.; Li, L.-J.; Li, K.; and Fei-Fei, L. 2009. ImageNet: A Large-Scale Hierarchical Image Database. In Proc. CVPR.

Devries, T.; and Taylor, G. W. 2017. Improved Regularization of Convolutional Neural Networks with Cutout. arXiv:1708.04552.

Diegterich, T. G. 2000. Ensemble Methods in Machine Learning. In Multiple Classifier Systems, 1–15.

Dong, X.; and Yang, Y. 2019. Searching for a Robust Neural Architecture in Four GPU Hours. In Proc. CVPR, 1761–1770.

Fort, S.; Hu, H.; and Lakshminarayanan, B. 2019. Deep ensembles: A loss landscape perspective. arXiv:1912.02757.

Gal, Y.; and Ghahramani, Z. 2016. Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning. In Proc. ICML, 1050–1059.

Goodfellow, I. J.; Shlens, J.; and Szegedy, C. 2015. Explaining and Harnessing Adversarial Examples. In Proc. ICLR.

Guo, Z.; Zhang, X.; Mu, H.; Heng, W.; Liu, Z.; Wei, Y.; and Sun, J. 2020. Single Path One-Shot Neural Architecture Search with Uniform Sampling. In Proc. ECCV, 544–560.

Hong, W.; Li, G.; Zhang, W.; Tang, R.; Wang, Y.; Li, Z.; and Yu, Y. 2020. DropNAS: Grouped Operation Dropout for Differentiable Architecture Search. In Proc. IJCAI, 2326–2332.

Huang, G.; Li, Y.; Pleiss, G.; Liu, Z.; Hopcroft, J. E.; and Weinberger, K. Q. 2017a. Snapshot Ensembles: Train 1, Get M for Free. In Proc. ICLR.

Huang, G.; Liu, Z.; van der Maaten, L.; and Weinberger, K. Q. 2017b. Densely Connected Convolutional Networks. In Proc. CVPR, 2261–2269.

Jang, E.; Gu, S.; and Poole, B. 2017. Categorical Reparameterization with Gumbel-Softmax. In Proc. ICLR.

Kingma, D. P.; and Ba, J. 2015. Adam: A Method for Stochastic Optimization. In Proc. ICLR.

Kingma, D. P.; and Welling, M. 2014. Auto-Encoding Variational Bayes. In Proc. ICLR.

Krizhevsky, A.; Hinton, G.; et al. 2009. Learning multiple layers of features from tiny images. Technical report, Citeseer.

Lakshminarayanan, B.; Pritzel, A.; and Blundell, C. 2017. Simple and Scalable Predictive Uncertainty Estimation using Deep Ensembles. In Proc. NIPS, 6402–6413.

Liu, C.; Zoph, B.; Neumann, M.; Shlens, J.; Hua, W.; Li, L.; Fei-Fei, L.; Yuille, A. L.; Huang, J.; and Murphy, K. 2018. Progressive Neural Architecture Search. In Proc. ECCV, 19–35.

Liu, H.; Simonyan, K.; and Yang, Y. 2019. DARTS: Differentiable Architecture Search. In Proc. ICLR.

Liu, Q.; and Wang, D. 2016. Stein Variational Gradient Descent: A General Purpose Bayesian Inference Algorithm. In Proc. NIPS, 2370–2378.

Maddison, C. J.; Mnih, A.; and Teh, Y. W. 2017. The Concrete Distribution: A Continuous Relaxation of Discrete Random Variables. In Proc. ICLR.

Madry, A.; Makelov, A.; Schmidt, L.; Tsipras, D.; and Vladu, A. 2018. Towards Deep Learning Models Resistant to Adversarial Attacks. In Proc. ICLR.

Parikh, N.; and Boyd, S. P. 2014. Proximal Algorithms. Found. Trends Optim., 1(3): 127–239.

Pham, H.; Guan, M. Y.; Zoph, B.; Le, Q. V.; and Dean, J. 2018. Efficient Neural Architecture Search via Parameter Sharing. In Proc. ICML, 4092–4101.

Real, E.; Aggarwal, A.; Huang, Y.; and Le, Q. V. 2019. Regularized Evolution for Image Classifier Architecture Search. In Proc. AAAI, 4780–4789.

Snoek, J.; Larochelle, H.; and Adams, R. P. 2012. Practical Bayesian Optimization of Machine Learning Algorithms. In Proc. NIPS, 2960–2968.

Strauss, T.; Hanselmann, M.; Junghering, A.; and Ulmer, H. 2017. Ensemble methods as a defense to adversarial perturbations against deep neural networks. arXiv:1709.03423.

Van der Maaten, L.; and Hinton, G. 2008. Visualizing data using t-SNE. J. Mach. Learn. Res., 9(11).

Xie, S.; Zheng, H.; Liu, C.; and Lin, L. 2019. SNAS: Stochastic neural architecture searching. In Proc. ICLR.

Xu, Y.; Xie, L.; Zhang, X.; Chen, X.; Qi, G.; Tian, Q.; and Xiong, H. 2020. PC-DARTS: Partial Channel Connections for Memory-Efficient Architecture Search. In Proc. ICLR.

Zaidi, S.; Zela, A.; Elsken, T.; Holmes, C.; Hutter, F.; and Teh, Y. W. 2020. Neural ensemble search for performant and calibrated predictions. arXiv:2006.08573.

Zhou, Z.-H. 2012. Ensemble methods: Foundations and algorithms. CRC press.

Zhuo, J.; Liu, C.; Shi, J.; Zhu, J.; Chen, N.; and Zhang, B. 2018. Message Passing Stein Variational Gradient Descent. In Proc. ICML, 6013–6022.

Zoph, B.; and Le, Q. V. 2017. Neural Architecture Search with Reinforcement Learning. In Proc. ICLR.

Zoph, B.; Vasudevan, V.; Shlens, J.; and Le, Q. V. 2018. Learning Transferable Architectures for Scalable Image Recognition. In Proc. CVPR, 8697–8710.
A Proofs

Proposition. (Training fairness in supernet.) Let $T$ and $T_A$ denote the number of steps applied to train the supernet and candidate architecture $A_i$ 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

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

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

$$X^t_i - X^t_j = \begin{cases} +1, & p = 1/N \\ 0, & p = (N - 2)/N \\ -1, & p = 1/N \end{cases}. \quad (11)$$

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

$$\Pr(\lim_{T \to \infty} T^{-1} \sum_{t=1}^{T} X^t_i - X^t_j = 0) = 1. \quad (12)$$

Note that

$$T^{-1}(T_{A_i} - T_{A_j}) = T^{-1} \sum_{t=1}^{T} X^t_i - X^t_j. \quad (13)$$

We thus can complete this proof by

$$\Pr(\lim_{T \to \infty} T^{-1}(T_{A_i} - T_{A_j}) = 0) = 1. \quad (14)$$

Proof of Proposition 1. As particles $\{x_i\}^n_{i=1}$ 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

$$n\delta\mathbb{E}_{x \sim q} [k(x, x')] \approx \delta/n \sum_{i=1}^{n} \sum_{j=1}^{n} k(x_i, x_j) \triangleq \sum_{i=1}^{n} h(x_i),$$

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

$$\nabla x_i \sum_{j=1}^{n} h(x_j) = \nabla x_i h(x_k). \quad (16)$$

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

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

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 $h(y^*)$ with its first-order Taylor expansion, i.e., $h(y^*) \approx h(x_i) + \nabla_x h(x_i)(y^* - x_i)$ and achieve following approximation:

$$\text{prox}_h(x^+_i) \approx x^+_i - \nabla h(x_i)$$

$$\approx x_i + \epsilon \phi^*(x_i) \approx x_i + \epsilon \phi^*(x_i) - \delta/n \sum_{j=1}^{n} \nabla_x h(x_i, x_j). \quad (19)$$

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

$$x_i \leftarrow x_i + 1/n \sum_{j=1}^{n} k(x_j, x_i) \nabla x_j \log p(x_j)$$

$$+ \nabla x_i h(x_j, x_i) - \delta \nabla x_i h(x_j, x_i). \quad (20)$$

Proof of Proposition 2. Notably, since $k(x, x') = c$ when $x = x'$, we will achieve a constant $k(x, x)$ for any particle $x$ 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 $x$ as

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

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

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

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

$$q^* = \arg \min_{q} \text{KL}(q||p)$$

$$= \arg \min_{x'} \{-\log p(x')\} \quad (23)$$

which finally concludes the proof.

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

B Experimental Settings

B.1 The Search Space

We use the DARTS (Liu, Simonyan, and Yang 2019) search space in our paper: Each candidate architecture in the search space consists of a stack of $L$ cells, which can be represented as a directed acyclic graph (DAG) of $N$ nodes denoted by
represented as $\{z_0, z_1, \ldots, z_{N-1}\}$. 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, Simonyan, and Yang (2019), to select the best-performing architectures, we need to select their corresponding cells, including the normal and reduction cell. We refer to DARTS (Liu, Simonyan, and Yang 2019) for more details. In practice, this search space is represented as a supernet stacked by 8 cells (6 normal cells and 2 reduction cells) with initial channels of 16.

### B.2 The 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 a fair and sufficient model training of each candidate architecture, we adopt stochastic gradient descent (SGD) with epoch 50, learning rate cosine scheduled from 0.1 to 0, momentum 0.9, weight decay $3 \times 10^{-4}$ and batch size 128 in the model training of the supernet, where one candidate architecture is uniformly randomly sampled from this supernet in each training step.

### B.3 Posterior Distribution

#### Variational posterior distribution.

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

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

where $\tau$ 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

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

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 (Jang, Gu, and Poole 2017; Maddison, Mnih, and Teh 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 $p_\alpha(A)$ to estimate $E_{A \sim p_\alpha(A)} [\log p(D|A)]$ (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 $3 \times 10^{-4}$ to update our variational posterior distribution $p_\alpha(A)$ 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 $p_\alpha(A)$ 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 $o \in O$ in (24) into a one-hot vector $h_o$. By introducing the random variable $\alpha_i \in \mathbb{R}^{|O|}$ and multi-variate normal distribution $N(\alpha_i|h_o, \Sigma)$ into our relaxation, our relaxed posterior distribution of neural architectures can be framed as

$$\tilde{p}_\alpha(A) = \prod_{i=2}^{N-2} 1/Z_i \sum_{o \in O} p_\alpha(o)N(\alpha_i|h_o, \Sigma), \quad (26)$$

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

$$o^*_i = \arg \min_{o \in O} \|o^*_i - h_o\|_2. \quad (27)$$

#### 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 $\Sigma$ in (26) is set to an identity matrix scaled by $|O|$. Besides, the parameter $\delta$ is optimized as a hyper-parameter via grid search or Bayesian Optimization (Snoek, Larochelle, and Adams 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 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, Simonyan, and Yang 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 $3 \times 10^{-4}$, 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, Simonyan, and Yang 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 (Chen et al. 2019; Liu, Simonyan, and Yang 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 $3 \times 10^{-5}$. 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 NESS algorithms by comparing the model robustness 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 robustness 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, Shlens, and Szegedy 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 $L_{\infty}$ norm constrain of 0.1. 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, $[0,0.001,10]$, 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 NESS algorithms. 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.

Table 4 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 be further justified given the search results in our Sec. 4.2.

| 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 4: The correlation between the estimated and true performances of candidate architectures and their ensembles in the DARTS search space on CIFAR-10 with varying ensemble size.
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.

### 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 2019; 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 2019; 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.

**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.

### Table 5: The comparison of true ensemble test error (%) on CIFAR-10 achieved by our NESS algorithms using the post-training posterior distribution and its best-response counterpart on CIFAR-10 with the ensemble size of n=3. We use Δ to denote the improved generalization performance achieved by our post-training posterior distribution.

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

**Table 6: The comparison of generalization performance with SOTA image classifiers on ImageNet. The ensemble size is set to n=3 for NES-RS and our NESS algorithms.**

| Architecture(s)      | Test Error (%) | Params | +x× |
|----------------------|----------------|--------|-----|
|                      | Top-1          | Top-5  | (M) | (M) |
| 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 |
| DARTS-               | 23.8           | 7.0    | 4.5 | 467 |
| SDARTS-ADV           | 25.2           | 7.8    | 5.4 | 594 |
| NES-RS               | 23.4           | 6.8    | 3.9 | 432† |
| NESS (MC Sampling)   | **22.3**       | **6.2**| **4.6†**| **522†** |
| NESS (SVGD-RD)       | **22.3**       | **6.1**| **4.9†**| **562†** |

† Reported as the average of parameter size or multiply-add operations using the neural architectures in an ensemble.

**Improved performance of selected ensembles using post-training posterior distribution.** We then compare the final ensemble test performance achieved by our NESS algorithms 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 5 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.
Figure 5: The comparison of search effectiveness (test error of ensembles, i.e., the y-axis) and efficiency (evaluation budget, i.e., the x-axis) between our NESS (MC Sampling) and NESS (SVGD-RD) algorithm under varying softmax temperature \( \tau \). The single best baseline refers to the single best architecture achieving the lowest test error in the search space. We evaluate the test error of each ensemble using the model parameters inherited from their supernet, and the test errors are reported with the mean and standard error of five independent trials.

Figure 6: The comparison of ensemble test error achieved by our NESS (SVGD-RD) algorithm with varying \( \delta \) under different softmax temperature \( \tau \). The test error for each \( \delta \) is reported with the mean and standard error of five independent trials.

C.3 Ensemble for Classification on ImageNet

Table 6 demonstrates the comparison of the generalization performance on ImageNet among the architectures (or the ensembles of neural architectures) selected by various NAS and ensemble search algorithms. We follow Appendix B.5 to evaluate our final selected ensembles that are obtained in Sec. 4.2 on ImageNet. As shown in Table 6, our NESS algorithms enjoy improved generalization performances when compared with conventional NAS algorithms and also an ensemble search algorithm (i.e., NES-RS), which is consistent with the results in Sec. 4.2. Overall, these results further validate the effectiveness of our NESS algorithms in selecting ensembles with improved generalization performance.

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 NESS (MC Sampling) and NESS (SVGD-RD) algorithm with varying softmax temperature \( \tau \) (appeared in (24)). We use these posterior distributions with varying temperature \( \tau \) to simulate the possible inaccurate posterior distributions we may obtain in practice. Figure 5 illustrates the comparison on CIFAR-10 in the DARTS search space with an ensemble size of \( n = 5 \). Notably, our NESS (SVGD-RD) with controllable diversity can consistently achieve improved search effectiveness and efficiency than our NESS (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 NESS (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 NESS (MC Sampling).

We further provide the comparison of ensemble test error achieved by our SVGD-RD with varying \( \delta \) under different softmax temperature \( \tau \) in Figure 6. Notably, when the posterior distribution tends to be flatter (i.e., \( \tau = 10 \)), a smaller \( \delta \) is preferred by our SVGD-RD in order to sample architectures with better single-model performances while maintaining the compelling predictive diversity. Meanwhile, when this posterior distribution tends to be sharper (i.e., \( \tau = 0.1 \)), a larger \( \delta \) is preferred by our SVGD-RD in order to sample architectures with a larger predictive diversity 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 predictive diversity, our SVGD-RD is thus capable of achieving comparable generalization performance under varying \( \tau \), which further validates the advantages of controllable diversity in our SVGD-RD.