Neural Ensemble Search via Bayesian Sampling

Yao Shu\(^1\)  Yizhou Chen\(^1\)  Zhongxiang Dai\(^1\)  Bryan Kian Hsiang Low\(^1\)

\(^1\)Department of Computer Science, National University of Singapore, Singapore

Abstract

Recently, \textit{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 \textit{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 \textit{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, thus indicating the superior performance of our NESBS algorithm over these 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 \textit{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., search cost) or the search effectiveness (i.e., 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 have thus 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, \textit{neural network ensembles} are widely known to be capable of achieving an improved performance compared with a single neural network in practice [Cortes et al., 2017; Gal and Ghahramani, 2016; Lakshminarayanan et al., 2017]. This naturally begs the question: \textit{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 \textit{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 will not be affordable in resource-constrained scenarios.

To this end, this paper introduces a novel algorithm, namely \textit{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,
i.e., single-model performances and diverse model predictions (Sec. 2.2). 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 [Real et al. 2019; Zoph and Le 2017; Zoph et al. 2018] 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; Chen and Hsieh 2020; Chu et al. 2020; Liu et al. 2019; Xie et al. 2019] 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 Monte 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 will not be affordable 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 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 \( p(x) \) with a simpler density \( q^*(x) \) in a predefined set \( Q \) by minimizing the Kullback-Leibler (KL) divergence between these two densities:

\[
q^* = \arg \min_{q \in Q} \text{KL}(q||p) \triangleq \mathbb{E}_q \left[ \log \left( \frac{q(x)}{p(x)} \right) \right].
\]

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 \( \phi^*(x_i) \) and a step size \( c \):

\[
x_i \leftarrow x_i + c\phi^*(x_i) \quad \text{for } i = 1, \ldots, n.
\]

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) \( \mathcal{H} \triangleq \mathcal{H}_0 \times \cdots \times \mathcal{H}_0 \) 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:

\[
\phi^* = \arg \max_{\phi \in \mathbb{F}} \left\{ -\frac{d}{d\epsilon} \text{KL}(q_{[\epsilon \phi]} || p) \bigg|_{\epsilon=0} \right\},
\]

which yields a closed-form solution:

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

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

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

As revealed in Liu and Wang [2016], the two terms in the aforementioned closed-form solution take different effects: The first term with \( \nabla_{x_j} \log p(x) \) 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

Contrary 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 \( f_A(x, \theta_A) \) denote the output of an architecture \( A \) with input data \( x \) and model parameter \( \theta_A \). \( S \) be a set of architectures, \( \Theta_S \) be a set of the corresponding model parameters of these architectures, and \( \mathcal{L}_{\text{train}} \) and \( \mathcal{L}_{\text{val}} \) denote the training and validation losses, respectively. Given the ensemble scheme \( \mathcal{F}_S(x, \Theta_S) \triangleq n^{-1} \sum_{A \in S} f_A(x, \theta_A) \) with an ensemble size of \( |S| = n \), NES can be formally framed as

\[
\min_S \mathcal{L}_{\text{val}}(\mathcal{F}_S(x, \Theta_S)) \\
\text{s.t. } \forall \theta^*_A \in \Theta^*_S, \theta^*_A = \arg \min \mathcal{L}_{\text{train}}(f_A(x, \theta_A)). \tag{6}
\]

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., lower-level optimization in (6)) unaffordable. (II) The ensemble search space is exponentially increasing in the ensemble size \( n \): For example, there are \( \sim m^n \) different ensembles given \( m \) diverse architectures. The combinatorial optimization problem (i.e., 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. Thus, 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 novelty 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.

3.1 MODEL TRAINING OF SUPERNET

Similar to one-shot NAS algorithms [Liu et al., 2019, Pham et al., 2018], 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 Fig. 1). The training fairness of such a training scheme can then be theoretically guaranteed, as demonstrated in 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

---

The text above is the extracted content from the image. It is a detailed explanation of the neural ensemble search via Bayesian sampling, discussing the challenges and the proposed solution in a structured manner. The text includes mathematical formulations and algorithms, with references and examples to illustrate the concepts. The primary focus is on the model training of a supernet and the exploration of the ensemble search space. The text is structured to provide a clear understanding of the problem and the proposed solution.
factors. Let \( \mathcal{D} \) denote the validation dataset, and \( p(\mathcal{A}) \) and \( p(\mathcal{A}|\mathcal{D}) \) denote, respectively, 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,

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

(7) where \( p(\mathcal{D}|\mathcal{A}) \) (i.e., 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 \( \gamma \)-Lipschitz continuous loss function \( \mathcal{L}(\mathcal{f}) \), the diversity of model predictions (i.e., \( \| \mathbf{f}_{\mathcal{A}_1} - \mathbf{f}_{\mathcal{A}_2} \|_2 \) can then be lower bounded based on the Lipschitz continuity of \( \mathcal{L}(\cdot) \):

\[
\| \mathbf{f}_{\mathcal{A}_1} - \mathbf{f}_{\mathcal{A}_2} \|_2 \geq \gamma^{-1} \mathcal{L}(\mathbf{f}_{\mathcal{A}_1}) - \mathcal{L}(\mathbf{f}_{\mathcal{A}_2}) .
\]

(8) Therefore, (8) suggests that in addition to being able to characterize the single-model performances of architectures (i.e., \( \mathcal{L}(\mathcal{f}) \)), the posterior distribution \( p(\mathcal{A}|\mathcal{D}) \) can 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 \( p_\alpha(\mathcal{A}) \) (parameterized by a low-dimensional \( \alpha \)) that can be optimized via variational inference, i.e., by minimizing the KL divergence between \( p_\alpha(\mathcal{A}) \) 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 \( p_\alpha(\mathcal{A}) \):

\[
\max_{\alpha} \mathbb{E}_{\mathcal{A} \sim p_\alpha(\mathcal{A})} [\log p(\mathcal{D}|\mathcal{A})] - \text{KL}[p_\alpha(\mathcal{A})||p(\mathcal{A})] .
\]

(9) Similar to [Kingma and Welling, 2014], a 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 Appendix C.2.

### 3.3 BAYESIAN SAMPLING

To solve (9) 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].

#### 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

---

**Algorithm 1 NES via Bayesian Sampling (NESBS)**

1: **Input**: Iterations \( T \), ensemble size \( n \), a supernet \( \mathcal{A}_0 \)
2: Train the supernet to get its tuned parameters \( \theta^* \)
3: Obtain the posterior distribution \( p_\alpha^*(\mathcal{A}) \) with (9)
4: for iteration \( t = 1, \ldots, T \) do
5: Sample \( S_t \) of size \( n \) via Algorithm 2 or 3
6: Evaluate estimated \( \mathcal{L}(\mathcal{F}_{S_t}(x, \Theta_{S_t})) \) given \( \theta^* \)
7: end for
8: Select optimum \( S^* = \arg \min_{S_t} \mathcal{L}(\mathcal{F}_{S_t}(x, \Theta_{S_t}^*)) \)

**Algorithm 2 MC Sampling**

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

**Algorithm 3 SVGD-RD**

1: **Input**: Diversity coefficient \( \delta \), ensemble size \( n \), iterations \( L \), initial particles \( \{ x_i^{(0)} \}_{i=1}^n \), posterior \( p_\alpha^*(\mathcal{A}) \), kernel \( k(x, x') \), step size \( \{ \epsilon_t \}_{t=1}^T \)
2: for iteration \( l = 0, \ldots, L - 1 \) do
3: Evaluate updates \( \hat{\phi}_l(x) = \frac{1}{n} \sum_{j=1}^n \nabla x_j k(x_j^{(l)}, x) - \delta \nabla x k(x_j^{(l)}, x) + k(x_j^{(l)}, x) \nabla x_j \log p_\alpha^* \)
4: Update particles \( x_i^{(l+1)} \leftarrow x_i^{(l)} + \epsilon_t \hat{\phi}_l(x_i^{(l)}) \)
5: end for
6: **Output**: \( S = \{ \mathcal{A}_i \}_{i=1}^n \) derived based on \( \{ x_i^{(L)} \}_{i=1}^n \)
Figure 2: Impact of $\delta$ in SVGD-RD. We use contours and dots to denote the density of target distribution and sampled particles, respectively. The target distribution is chosen to be $p(x) = (1/Z) \left[ N(x|u_1, \Sigma_1) + N(x|u_2, \Sigma_2) \right]$ where $u_1 = (-1, 0)$, $u_2 = (0, 1)$, $\Sigma_1 = \Sigma_2 = \text{diag}(0.25, 0.5)$, and $Z$ denotes the normalization constant. Those sampled particles are obtained from Algorithm 3 using $L = 1000$, $n = 15$, $\epsilon_l = 0.1$, and a radial basis function (RBF) kernel. Notably, SVGD-RD tends to sample particles with more diverse probability densities as $\delta$ is increased, hence indicating a controllable diversity (via $\delta$) in our SVGD-RD algorithm. Meanwhile, SVGD-RD can consistently sample particles with high probability densities under varying $\delta$.

compelling single-model performances.

3.3.2 SVGD with Regularized Diversity (SVGD-RD)

However, the diversity of sampled architectures using the MC Sampling algorithm above cannot be controlled and hence may 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 (9)). 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 still cannot be controlled by human experts.

We hence develop an SVGD with regularized diversity (SVGD-RD) sampling algorithm that can achieve a controllable diversity among those sampled particles. We follow the notations from Sec. 3.2. In particular, when optimizing the distribution $q^*$ (represented by the $n$ particles $\{x_i^*\}_{i=1}^n$), we modify the objective in (10) by adding a term representing the (controllable) diversity among the particles measured by the kernel function $k(x, x')$:

$$q^* = \arg \min_{q \in \Omega} \text{KL}(q || p) + n \delta \mathbb{E}_{x, x' \sim q} [k(x, x')] \quad (10)$$

where $\delta$ is the parameter explicitly controlling the diversity, and $p$ in $\{x_i^*\}_{i=1}^n$ denotes the posterior distribution $p_{x^*}(.|A)$ derived in Sec. 3.2 which we intend to sample from. Following the work of SVGD, $q^*$ in (10) is represented by $\{x_i^*\}_{i=1}^n$ 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. 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 its optimization details in practice. After obtaining those optimal particles $\{x_i^*\}_{i=1}^n$ 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 $\text{prox}_\delta(y) = \arg \min_z \|z - y\|_2^2$, by applying proximal gradient method [Parikh and Boyd, 2014] and proper approximation, (10) can be optimized via the following updates of the particles $\{x_i\}_{i=1}^n$:

$$x_i \leftarrow x_i - \frac{1}{n} \sum_{j=1}^n k(x_j, x_i) \nabla_{x_i} \log p(x_j)$$

$$+ \nabla_{x_i} k(x_j, x_i) - \delta \nabla_{x_i} k(x_j, x_i)$$

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 1 below (its proof is in Appendix A). On the other hand, the maximization of the scaled distance $-n \delta \mathbb{E}_{x, x' \sim q} [k(x, x')]$ among the sampled particles leads to a controllable diversity (via $\delta$) among these sampled particles and also a controllable diversity of the probability densities among these particles (see Fig. 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 $k(x, x') = c$ for every $x = x'$ 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. $x$ in the case of $n = 1$.

4 EXPERIMENTS

4.1 SEARCH IN NAS-BENCH-201

To verify the effectiveness and efficiency of our NESBS algorithm, we firstly compare it with other well-known
We further demonstrate the superior search effectiveness and whereas a small number of training epochs can already contribute to well-performing results for our NESBS algorithm. This is likely because more training epochs have been used in these 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.

**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., 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.5.

**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.

**Ensemble for adversarial defense.** Ensemble methods have already been shown to be an essential and effective de-
Table 2: Comparison of different image classifiers on CIFAR-10/100. Results of MC DropPath are from a drop probability of 0.01 and our search costs are evaluated on Nvidia 1080Ti.

<table>
<thead>
<tr>
<th>Architecture(s)</th>
<th>Test Error (%)</th>
<th>Params (M)</th>
<th>Search Cost (GPU Days)</th>
<th>Search Method</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>C10</td>
<td>C100</td>
<td>C10</td>
<td>C100</td>
</tr>
<tr>
<td><strong>NAS algorithms</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NASNet-A [Zoph et al., 2018]</td>
<td>2.65</td>
<td>-</td>
<td>3.3</td>
<td>-</td>
</tr>
<tr>
<td>AmoebaNet-A [Real et al., 2019]</td>
<td>3.34</td>
<td>18.93</td>
<td>3.2</td>
<td>3.1</td>
</tr>
<tr>
<td>PNAS [Liu et al., 2018]</td>
<td>3.41</td>
<td>19.53</td>
<td>3.2</td>
<td>3.2</td>
</tr>
<tr>
<td>ENAS [Pham et al., 2018]</td>
<td>2.89</td>
<td>19.43</td>
<td>4.6</td>
<td>4.6</td>
</tr>
<tr>
<td>DARTS [Liu et al., 2019]</td>
<td>2.76</td>
<td>17.54</td>
<td>3.3</td>
<td>3.4</td>
</tr>
<tr>
<td>GDAS [Dong and Yang, 2019a]</td>
<td>2.93</td>
<td>18.38</td>
<td>3.4</td>
<td>3.4</td>
</tr>
<tr>
<td>P-DARTS [Chen et al., 2019]</td>
<td>2.50</td>
<td>-</td>
<td>3.4</td>
<td>-</td>
</tr>
<tr>
<td>DARTS- (avg) [Chu et al., 2020]</td>
<td>2.59</td>
<td>17.51</td>
<td>3.5</td>
<td>3.3</td>
</tr>
<tr>
<td>SDARTS-ADV [Chen and Hsieh, 2020]</td>
<td>2.61</td>
<td>-</td>
<td>3.3</td>
<td>-</td>
</tr>
<tr>
<td><strong>Ensemble (search) algorithms</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MC DropPath (ENAS)</td>
<td>2.88</td>
<td>16.83</td>
<td>3.8(^\dagger)</td>
<td>3.9(^\dagger)</td>
</tr>
<tr>
<td>DeepEns (ENAS)</td>
<td>2.49</td>
<td>15.04</td>
<td>3.8(^\dagger)</td>
<td>3.9(^\dagger)</td>
</tr>
<tr>
<td>DeepEns (DARTS)</td>
<td>2.42</td>
<td>14.56</td>
<td>3.3(^\dagger)</td>
<td>3.4(^\dagger)</td>
</tr>
<tr>
<td>NES-RS(^\dagger) [Zaidi et al., 2021]</td>
<td>2.50</td>
<td>15.24</td>
<td>3.0(^\dagger)</td>
<td>3.1(^\dagger)</td>
</tr>
<tr>
<td><strong>Our ensemble search algorithm</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NESBS (MC Sampling)</td>
<td>2.41</td>
<td>14.70</td>
<td>3.8(^\dagger)</td>
<td>3.9(^\dagger)</td>
</tr>
<tr>
<td>NESBS (SVGD-RD)</td>
<td>2.36</td>
<td>14.55</td>
<td>3.7(^\dagger)</td>
<td>3.8(^\dagger)</td>
</tr>
</tbody>
</table>

\(^\dagger\) Reported as the averaged parameter size of the architectures in a neural network ensemble.

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

<table>
<thead>
<tr>
<th>Architecture(s)</th>
<th>Test Error (%)</th>
<th>Params (M)</th>
<th>+×</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Top-1</td>
<td>Top-5</td>
<td></td>
</tr>
<tr>
<td><strong>NAS algorithms</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NASNet-A</td>
<td>26.0</td>
<td>8.4</td>
<td>5.3</td>
</tr>
<tr>
<td>AmoebaNet-A</td>
<td>25.5</td>
<td>8.0</td>
<td>5.1</td>
</tr>
<tr>
<td>PNAS</td>
<td>25.8</td>
<td>8.1</td>
<td>5.1</td>
</tr>
<tr>
<td>DARTS</td>
<td>26.7</td>
<td>8.7</td>
<td>4.7</td>
</tr>
<tr>
<td>GDAS</td>
<td>26.0</td>
<td>8.5</td>
<td>5.3</td>
</tr>
<tr>
<td>P-DARTS</td>
<td>24.4</td>
<td>7.4</td>
<td>4.9</td>
</tr>
<tr>
<td>SDARTS-ADV</td>
<td>25.2</td>
<td>7.8</td>
<td>5.4</td>
</tr>
<tr>
<td><strong>Ensemble (search) algorithm</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NES-RS</td>
<td>23.4</td>
<td>6.8</td>
<td>3.9</td>
</tr>
<tr>
<td><strong>Our ensemble search algorithm</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NESBS (MC Sampling)</td>
<td>22.3</td>
<td>6.2</td>
<td>4.6</td>
</tr>
<tr>
<td>NESBS (SVGD-RD)</td>
<td>22.3</td>
<td>6.1</td>
<td>4.9</td>
</tr>
</tbody>
</table>

Table 4 summarizizes the comparison of adversarial defense among ensemble (search) algorithms on CIFAR-10/100 under different white-box adversarial attacks, including the Fast Gradient Signed Method (FGSM) attack [Goodfellow et al., 2015], the Projected Gradient Descent (PGD) attack [Madry et al., 2018], the Carlini Wagner (CW) attack [Carlini and Wagner, 2017], and the AutoAttack [Croce and Hein, 2020]. 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. Besides, even regarding the adversarial robustness of the single models in an ensemble, the architectures selected by our NESBS are also more advanced (i.e., by achieving higher test accuracy in the Attack columns) than well-known architectures such as RobNet [Guo et al., 2020] and DARTS.
We demonstrate that the effectiveness of our NESBS is better achieved by 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 $n = 3$.

### Table 4: Comparison of adversarial defense among different ensemble (search) algorithms on CIFAR-10/100 under white-box adversarial attacks.

<table>
<thead>
<tr>
<th>Method</th>
<th>FGSM</th>
<th>PGD-40</th>
<th>CW</th>
<th>AutoAttack</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Attack (%)</td>
<td>Defense (%)</td>
<td>Attack (%)</td>
<td>Defense (%)</td>
</tr>
<tr>
<td>DeepEns</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>→ RobNet-free</td>
<td>66.62 ± 0.32</td>
<td>85.25 ± 0.39</td>
<td>41.81 ± 0.80</td>
<td>77.48 ± 0.67</td>
</tr>
<tr>
<td>→ ENAS</td>
<td>77.85 ± 0.58</td>
<td>87.94 ± 0.21</td>
<td>59.51 ± 1.13</td>
<td>86.57 ± 0.15</td>
</tr>
<tr>
<td>→ DARTS</td>
<td>76.79 ± 0.80</td>
<td>88.21 ± 0.14</td>
<td>57.71 ± 1.65</td>
<td>82.02 ± 0.10</td>
</tr>
<tr>
<td>NES-RS</td>
<td>79.19 ± 1.39</td>
<td>89.32 ± 0.27</td>
<td>65.59 ± 2.11</td>
<td>85.22 ± 0.41</td>
</tr>
<tr>
<td>NESBS (MC Sampling)</td>
<td>78.75 ± 1.29</td>
<td>89.15 ± 0.08</td>
<td>63.60 ± 1.87</td>
<td>85.35 ± 0.31</td>
</tr>
<tr>
<td>NESBS (SVGD-RD)</td>
<td>79.12 ± 0.61</td>
<td>89.86 ± 0.33</td>
<td>65.53 ± 1.56</td>
<td>85.37 ± 0.38</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DeepEns</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>→ RobNet-free</td>
<td>36.47 ± 0.25</td>
<td>61.39 ± 0.30</td>
<td>18.18 ± 0.47</td>
<td>52.61 ± 0.13</td>
</tr>
<tr>
<td>→ ENAS</td>
<td>46.40 ± 0.37</td>
<td>64.94 ± 0.27</td>
<td>28.87 ± 0.27</td>
<td>56.79 ± 0.25</td>
</tr>
<tr>
<td>→ DARTS</td>
<td>46.98 ± 0.57</td>
<td>65.38 ± 0.23</td>
<td>28.78 ± 0.74</td>
<td>57.10 ± 0.04</td>
</tr>
<tr>
<td>NES-RS</td>
<td>47.10 ± 1.46</td>
<td>65.33 ± 0.36</td>
<td>30.68 ± 1.66</td>
<td>58.80 ± 0.80</td>
</tr>
<tr>
<td>NESBS (MC Sampling)</td>
<td>50.69 ± 1.58</td>
<td>67.63 ± 0.05</td>
<td>33.37 ± 0.42</td>
<td>60.36 ± 0.62</td>
</tr>
<tr>
<td>NESBS (SVGD-RD)</td>
<td>51.47 ± 0.40</td>
<td>66.66 ± 0.13</td>
<td>35.02 ± 0.37</td>
<td>59.96 ± 0.18</td>
</tr>
</tbody>
</table>

### 4.3 SINGLE-MODEL PERFORMANCES AND DIVERSE MODEL PREDICTIONS

We demonstrate that the effectiveness of our NESBS results 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) 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 ATE of the models in their ensembles and a t-SNE plot of their model predictions, respectively.

Table 5 and Fig. 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]. Thus, Table 5 and Fig. 3 provide empirical justifications for the improved effectiveness of NESBS.

### Table 5: 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 on CIFAR-10/100.

<table>
<thead>
<tr>
<th>Method</th>
<th>C10</th>
<th>C100</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ATE</td>
<td>PPD</td>
</tr>
<tr>
<td>MC DropPath (DARTS)</td>
<td>2.71</td>
<td>0.39</td>
</tr>
<tr>
<td>DeepEns (DARTS)</td>
<td>2.69</td>
<td>2.08</td>
</tr>
<tr>
<td>NES-RS</td>
<td>2.87</td>
<td>2.29</td>
</tr>
<tr>
<td>NESBS (MC Sampling)</td>
<td>2.80</td>
<td>2.57</td>
</tr>
<tr>
<td>NESBS (SVGD-RD)</td>
<td>2.78</td>
<td>2.27</td>
</tr>
</tbody>
</table>

### 5 CONCLUSION

This paper presents a novel neural ensemble search algorithm, 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 performance of our NESBS in practice.
### Acknowledgements

This research/project is supported by A*STAR under its RIE2020 Advanced Manufacturing and Engineering (AME) Programmatic Funds (Award A20H6b0151).

### References


