Neural Ensemble Search via Bayesian Sampling (Supplementary Material)

Yao Shu[1] **Yizhou Chen**[1] **Zhongxiang Dai**[1] **Bryan Kian Hsiang Low**[1]

[1]Department of Computer Science, National University of Singapore, Singapore

A PROOFS

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

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

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

$$X_i^t - X_j^t = \begin{cases} +1, & p = 1/N \\ 0, & p = (N-2)/N \\ -1, & p = 1/N \end{cases} . \tag{S1}$$

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

$$\Pr(\lim_{T\to\infty} T^{-1} \sum_{t=1}^{T} X_i^t - X_j^t = 0) = 1 . \tag{S2}$$

Note that

$$T^{-1}(T_{\mathcal{A}_i} - T_{\mathcal{A}_j}) = T^{-1} \sum_{t=1}^{T} X_i^t - X_j^t . \tag{S3}$$

We thus can complete this proof by

$$\Pr(\lim_{T\to\infty} T^{-1}(T_{\mathcal{A}_i} - T_{\mathcal{A}_j}) = 0) = 1 . \tag{S4}$$

□

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

$$n\delta\mathbb{E}_{\boldsymbol{x},\boldsymbol{x}'\sim q}\left[k(\boldsymbol{x}, \boldsymbol{x}')\right] \approx \delta/n \sum_{i=1}^{n}\sum_{j=1}^{n} k(\boldsymbol{x}_i, \boldsymbol{x}_j) \triangleq \sum_{i=1}^{n} h(\boldsymbol{x}_i) , \tag{S5}$$

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

$$\nabla_{\boldsymbol{x}_k} \sum_{i=1}^{n} h(\boldsymbol{x}_i) = \nabla_{\boldsymbol{x}_k} h(\boldsymbol{x}_k) . \tag{S6}$$

Let $\boldsymbol{x}_i^+ \triangleq \boldsymbol{x}_i + \epsilon\boldsymbol{\phi}^*(\boldsymbol{x}_i)$ $(\forall i \in \{1, \cdots, n\})$ denote the functional gradient decent in the RKHS $\mathcal{H}$ to minimize the KL divergence term in our (10). Based on (S6) above, given proximal operator $\text{prox}_h(\boldsymbol{x}^+) = \arg\min_{\boldsymbol{y}} h(\boldsymbol{y}) + 1/2\|\boldsymbol{y} - \boldsymbol{x}^+\|_2^2$, by using proximal gradient method Parikh and Boyd [2014], our (10) can then be optimized via the following update to each particle $\boldsymbol{x}_i$:

$$\boldsymbol{x}_i \leftarrow \text{prox}_h(\boldsymbol{x}_i^+) = \arg\min_{\boldsymbol{y}} h(\boldsymbol{y}) + 1/2\|\boldsymbol{y} - \boldsymbol{x}_i^+\|_2^2 . \tag{S7}$$

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

$$\text{prox}_h(\boldsymbol{x}_i^+) = \boldsymbol{y}^* = \boldsymbol{x}_i^+ - \nabla_{\boldsymbol{y}^*} h(\boldsymbol{y}^*) . \tag{S8}$$

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

$$\begin{aligned} \text{prox}_h(\boldsymbol{x}_i^+) &\approx \boldsymbol{x}_i^+ - \nabla_{\boldsymbol{x}_i} h(\boldsymbol{x}_) \\ &\approx \boldsymbol{x}_i + \epsilon\boldsymbol{\phi}^*(\boldsymbol{x}_i) - \nabla_{\boldsymbol{x}_i} h(\boldsymbol{x}_i) \\ &\approx \boldsymbol{x}_i + \epsilon\boldsymbol{\phi}^*(\boldsymbol{x}_i) - \delta/n \textstyle\sum_j^n \nabla_{\boldsymbol{x}_i} k(\boldsymbol{x}_i, \boldsymbol{x}_j) . \end{aligned} \tag{S9}$$

Accepted for the 38th Conference on Uncertainty in Artificial Intelligence (UAI 2022).

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

$$\begin{aligned}\boldsymbol{x}_i \leftarrow \boldsymbol{x}_i + 1/n \textstyle\sum_{j=1}^{n} k(\boldsymbol{x}_j, \boldsymbol{x}_i) \nabla_{\boldsymbol{x}_j} \log p(\boldsymbol{x}_j) \\ + \nabla_{\boldsymbol{x}_j} k(\boldsymbol{x}_j, \boldsymbol{x}_i) - \delta \nabla_{\boldsymbol{x}_i} k(\boldsymbol{x}_j, \boldsymbol{x}_i) \; .\end{aligned} \tag{S10}$$

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

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

$$q(\boldsymbol{x}) = \begin{cases} 1 & \boldsymbol{x} = \boldsymbol{x}' \\ 0 & \boldsymbol{x} \neq \boldsymbol{x}' \end{cases} \; . \tag{S11}$$

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

$$\mathrm{KL}(q \| p) = \mathbb{E}_{q(\boldsymbol{x})}[\log(q(\boldsymbol{x})/p(\boldsymbol{x}))] = -\log p(\boldsymbol{x}') \; . \tag{S12}$$

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

$$\begin{aligned} q^* &= \arg\min_{q} \mathrm{KL}(q \| p) \\ &= \arg\min_{\boldsymbol{x}'} \{-\log p(\boldsymbol{x}')\} \\ &= \arg\max_{\boldsymbol{x}'} p(\boldsymbol{x}') \; , \end{aligned} \tag{S13}$$

which finally concludes the proof.

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

B EXPERIMENTAL SETTINGS

B.1 THE DARTS SEARCH SPACE

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

B.2 MODEL TRAINING OF SUPERNET

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

B.3 POSTERIOR DISTRIBUTION

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

$$p_{\boldsymbol{\alpha}_i}(o) = \frac{\exp(\alpha_i^o/\tau)}{\sum_{o \in \mathcal{O}} \exp(\alpha_i^o/\tau)} \; , \tag{S14}$$

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

$$p_{\boldsymbol{\alpha}}(A) = \prod_{i=2}^{N-2} p_{\boldsymbol{\alpha}_i}(o) \; . \tag{S15}$$

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

Optimization details. To optimize (9), we firstly relax our variational posterior distribution to be differentiable using the Straight-Through (ST) Gumbel-Softmax [Maddison et al., 2017, Jang et al., 2017] with the reparameterization trick. More precisely, we propose a variant of ST Gumbel-Softmax outputting the double-path architectures in the DARTS search space. Then, we use stochastic

gradient-based algorithms to optimize (9) efficiently. In each optimization step, we sample one neural architecture from the distribution $p_{\boldsymbol{\alpha}}(\mathcal{A})$ to estimate $\mathbb{E}_{\mathcal{A}\sim p_{\boldsymbol{\alpha}}(\mathcal{A})}\left[\log p(\mathcal{D}|\mathcal{A})\right]$ (i.e., the commonly used Cross-Entropy loss). In practice, we use Adam [Kingma and Ba, 2015] with learning rate 0.01, $\beta_1 = 0.9$, $\beta_2 = 0.999$ and weight decay 3×10^{-4} to update our variational posterior distribution $p_{\boldsymbol{\alpha}}(\mathcal{A})$ for 20 epochs.

B.4 SVGD OF REGULARIZED DIVERSITY

Continuous relaxation of variational posterior distribution. Notably, SVGD [Liu and Wang, 2016] and also our SVGD-RD is applied for continuous distribution. Unfortunately, the variational posterior distribution $p_{\boldsymbol{\alpha}}(\mathcal{A})$ is discrete due to a discrete search space. To apply SVGD-RD, we firstly relax this discrete posterior into its continuous counterpart using a mixture of Gaussian distribution. Specifically, we represent each operation $o \in \mathcal{O}$ in (S14) into a one-hot vector $\boldsymbol{h}_o$. By introducing the random variable $\boldsymbol{o}_i \in \mathbb{R}^{|\mathcal{O}|}$ and multi-variate normal distribution $\mathcal{N}(\boldsymbol{o}_i|\boldsymbol{h}_o, \Sigma)$ into our relaxation, our relaxed posterior distribution of neural architectures can be framed as

$$\widehat{p}_{\boldsymbol{\alpha}}(\mathcal{A}) = \prod_{i=2}^{N-2} 1/Z_i \sum_{o\in\mathcal{O}} p_{\boldsymbol{\alpha}_i}(o)\mathcal{N}(\boldsymbol{o}_i|\boldsymbol{h}_o, \Sigma) \;, \tag{S16}$$

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

$$o_i^* = \arg\min_{o\in\mathcal{O}} \|\boldsymbol{o}_i^* - \boldsymbol{h}_o\|_2 \;. \tag{S17}$$

Optimization details. Since Liu and Wang [2016] have demonstrated that SVGD is able to handle unnormalized target distributions, the normalization constant in (S16) can then be ignored in our SVGD-RD algorithm. In practice, the covariance matrix Σ in (S16) is set to an identity matrix scaled by $|\mathcal{O}|$. Besides, the parameter δ is optimized as a hyper-parameter via grid search or Bayesian Optimization [Snoek et al., 2012] within the range of $[-2, 1]$ in practice. To obtain well-performing particles in our SVGD-RD algorithms efficiently, we apply SGD using the gradient provided in Sec. 3.3.2 with a *radial basis function* (RBF) kernel on randomly initialized particles for L=1000 iterations under a learning rate of 0.1 and a momentum of 0.9.

B.5 EVALUATION ON BENCHMARK DATASETS

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

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

B.6 ADVERSARIAL DEFENSE

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

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

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 S1: The correlation between the estimated and true performances of candidate architectures and their ensembles in the DARTS search space on CIFAR-10.

when only a single model from an ensemble is required to be publicly available for model producers.

We apply the following attacks in our experiment: 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]. In both the FGSM attack and the PGD attack, we impose a L_∞ norm constrain of 0.01. The step size and the number of iterations in the PGD attack are set to 0.008 and 40, respectively. We adopt the same configurations of the CW attack under a L_2 norm constrain in [Carlini and Wagner, 2017]: We set the confidence constant, the range of constant c, the number of binary search steps, and the maximum number of optimization steps to 0, $[0.001, 10]$, 3, and 50, respectively; we then adopt Adam [Kingma and Ba, 2015] optimizer with learning rate 0.01 and $\beta_1 = 0.9$, $\beta_2 = 0.999$ in its search process. Besides, we adopt the same configuration of AutoAttack from [Croce and Hein, 2020].

C COMPLEMENTARY RESULTS

C.1 ENSEMBLE PERFORMANCE ESTIMATION

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

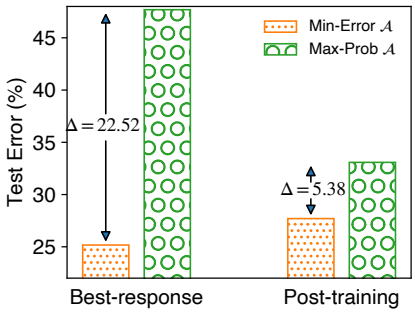

Figure S1: The comparison of performance discrepancy with the post-training and best-response posterior distribution on CIFAR-10. This performance discrepancy is measured by the gap of test error between the best-performing architecture (i.e., the architecture with the smallest test error) and the maximal-probability architecture (i.e., the architecture with the largest probability in the corresponding posterior distribution) in the DARTS search space.

Table S1 summarizes the results. Notably, the estimated and true performances are shown to be positively correlated in the case of n=$1, 3, 5$ by achieving relatively high Spearman and Pearson coefficients as well as a high agreement in these cases. Although the coefficients are low when the ensemble size is larger (i.e., n=7), the estimated and true performances are still capable of achieving a reasonably good agreement in this case. Based on these results, we argue that our estimated ensemble performance is informative and effective for our ensemble search. This effectiveness can also be supported by the competitive search results achieved by our NESBS in Sec. 4.2.

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

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

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

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

Table S2: The comparison of true ensemble test error (%) on CIFAR-10 achieved by our NESBS algorithm using the post-training posterior distribution and its best-response counterpart with an ensemble size of n=3. We use Δ to denote the improved generalization performance achieved by our post-training posterior distribution.

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 S1 illustrates the comparison. The results show that our post-training posterior distribution enjoys a smaller performance discrepancy, suggesting that our post-training posterior distribution is able to provide a more accurate characterization of the single-model performances. Interestingly, the best-response counterpart contributes to the best-performing architecture with a lower test error than our post-training posterior distribution, which should result from the Matthew Effect as justified in [Hong et al., 2020]. Specifically, well-performing architectures contribute to the frequent selections of these architectures for their model training during the optimization of the best-response posterior distribution. This will finally result in unfair model training in the search space and therefore the inaccurate characterization of single-model performances. Notably, we need a more accurate characterization of single-mode performance in this paper, as shown in Sec. 3.2. Therefore, our post-training posterior distribution should be more suitable than its best-response counterpart in our ensemble search.

Improved performance of selected ensembles using post–training posterior distribution. We then compare the final ensemble test performance achieved by our NESBS algorithm using the post-training posterior distribution and its best-response counterpart on CIFAR-10 with the ensemble size of $n = 3$. To obtain the final ensemble performance, we train each architecture in an ensemble for 100 epochs following the settings in Appendix B.5. Table S2 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.

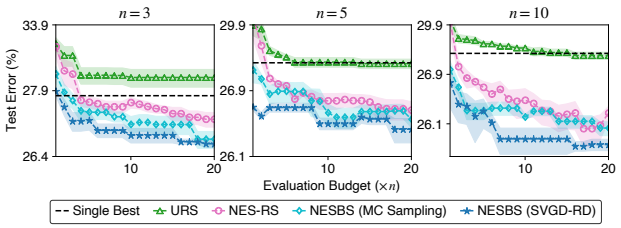

Figure S2: The comparison of search effectiveness (test error of ensembles in the y-axis) and efficiency (evaluation budget in the x-axis) for different ensemble search algorithms under varying ensemble size n. The single best baseline refers to the single best architecture achieving the lowest test error in the search space. The y-axis is shown in log-scale to ease visualization. Note that the test error for each algorithm is reported with the mean and standard error of five independent trials.

C.3 EFFECTIVENESS AND EFFICIENCY

As justified in Sec 3.3, both our MC Sampling and SVGD-RD algorithms can sample neural architectures with competitive single-model performances and diverse model predictions, which are known to be the criteria for well-performing ensembles [Zhou, 2012]. To further demonstrate that our algorithms are capable of selecting well-performing ensembles effectively and efficiently based on this sampling property, we compare our NESBS algorithm, including NESBS (MC Sampling) and NESBS (SVGD-RD), with the following ensemble search baselines on CIFAR-10 [Krizhevsky, 2009] in the DARTS [Liu et al., 2019] search space: (a) Uniform random sampling which we refer to as URS, and (b) NES-RS [Zaidi et al., 2021]. That is, we only replace the Bayesian sampling in our NESBS algorithm with these two different sampling/selection algorithms in this experiment and we keep using the model parameters inherited from a supernet to estimate the single-model and ensemble performances of architectures (including the test errors). The detailed experimental settings are in Appendix B.

Figure S2 illustrates the search results. Note that both NES-RS and our NESBS are able to achieve lower test errors than the single best-performing architecture in the search space. These results therefore demonstrate that these two ensemble search algorithms are indeed capable of achieving improved performance over conventional NAS algorithms that select only one single architecture from the search space. More importantly, given the same evaluation budgets, our NESBS algorithm consistently achieves lower test errors than URS and NES-RS, indicating the superior search effectiveness achieved by our NESBS algorithm. Meanwhile, our NESBS algorithm requires fewer evaluation budgets than URS and NES-RS to achieve comparable test errors, which also suggests that our algorithm is more efficient than URS and NES-RS. Interestingly, compared with MC Sampling, SVGD-RD

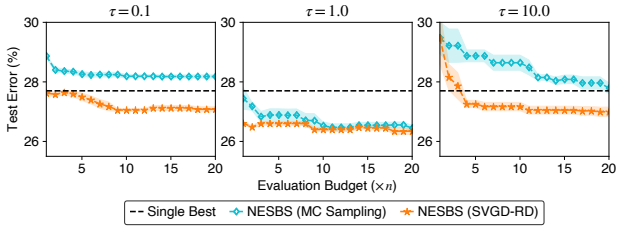

Figure S3: The comparison of search effectiveness (test error of ensembles in the y-axis) and efficiency (evaluation budget in the x-axis) between our NESBS (MC Sampling) and NESBS (SVGD-RD) algorithm under varying softmax temperature τ. The single best baseline refers to the single best architecture achieving the lowest test error in the search space. Each test error is reported with the mean and standard error of five independent trials.

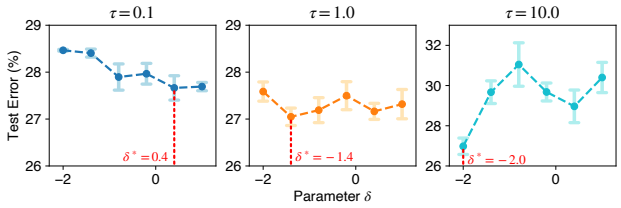

Figure S4: The comparison of ensemble test error achieved by our NESBS (SVGD-RD) algorithm with varying δ under different softmax temperature τ given an ensemble size of $n = 5$. We use δ^* to denote the optimal δ we obtained in our SVGD-RD algorithm under different temperature τ. The test error for each δ is reported with the mean and standard error of five independent trials.

can consistently produce improved search effectiveness and efficiency, which likely results from its controllable trade-off between the single-model performances and the diverse model predictions as justified in Sec. 3.3. Overall, these results have well justified the effectiveness and efficiency of our NESBS algorithm.

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

To examine the advantages of controllable diversity in our SVGD-RD, we firstly compare the search effectiveness and efficiency achieved by our NESBS (MC Sampling) and NESBS (SVGD-RD) algorithm with varying softmax temperature τ (appeared in (S14)). A larger temperature τ will lead to a flatter posterior distribution and hence degenerate its capability of characterizing single-model performances of neural architectures as indicated in (S14). We use these posterior distributions with varying temperature τ to simulate the possible posterior distributions we may obtain in practice. Figure S3 illustrates the comparison on CIFAR-10 in the DARTS search space with an ensemble size of $n = 5$. Notably, our NESBS (SVGD-RD) with controllable diversity can consistently achieve improved search effectiveness and efficiency than our NESBS (MC Sampling). Interestingly, this improvement becomes larger in the case of $\tau = 0.1, 10.0$, which should be the consequences of a bad exploration and exploitation achieved by our NESBS (MC Sampling), respectively. These results therefore suggest that the controllable diversity in our SVGD-RD generally can lead to improved search effectiveness and efficiency than our NESBS (MC Sampling).

We further provide the comparison of ensemble test error achieved by our SVGD-RD with varying δ under different softmax temperature τ in Figure S4. Notably, when the posterior distribution tends to be flatter (i.e., $\tau = 10$), a smaller δ is preferred by our SVGD-RD in order to sample architectures with better single-model performances while maintaining the compelling diverse model predictions. Meanwhile, when this posterior distribution tends to be sharper (i.e., $\tau = 0.1$), a larger δ is preferred by our SVGD-RD in order to sample architectures with a larger diverse model predictions while preserving the competitive single-model performances. Based on this controllable diversity and hence the controllable trade-off between the single-model performances and the diverse model predictions, our SVGD-RD is thus capable of achieving comparable performances under varying τ, which usually improve over our NESBS (MC Sampling) by comparing them with the results in Figure S3. These results further validate the advantages of the controllable diversity in our SVGD-RD.