Neural Ensemble Search for Uncertainty Estimation and Dataset Shift

Sheheryar Zaidi[1*] **Arber Zela**[2*] **Thomas Elsken**[3,2]
Chris Holmes[1] **Frank Hutter**[2,3] **Yee Whye Teh**[1]

[1]University of Oxford, [2]University of Freiburg, [3]Bosch Center for Artificial Intelligence
{szaidi, cholmes, y.w.teh}@stats.ox.ac.uk,
{zelaa, fh}@cs.uni-freiburg.de, thomas.elsken@de.bosch.com

Abstract

Ensembles of neural networks achieve superior performance compared to stand-alone networks in terms of accuracy, uncertainty calibration and robustness to dataset shift. *Deep ensembles*, a state-of-the-art method for uncertainty estimation, only ensemble random initializations of a *fixed* architecture. Instead, we propose two methods for automatically constructing ensembles with *varying* architectures, which implicitly trade-off individual architectures' strengths against the ensemble's diversity and exploit architectural variation as a source of diversity. On a variety of classification tasks and modern architecture search spaces, we show that the resulting ensembles outperform deep ensembles not only in terms of accuracy but also uncertainty calibration and robustness to dataset shift. Our further analysis and ablation studies provide evidence of higher ensemble diversity due to architectural variation, resulting in ensembles that can outperform deep ensembles, even when having weaker average base learners. To foster reproducibility, our code is available: https://github.com/automl/nes

1 Introduction

Some applications of deep learning rely only on point estimate predictions made by a neural network. However, many critical applications also require reliable predictive uncertainty estimates and robustness under the presence of dataset shift, that is, when the observed data distribution at deployment differs from the training data distribution. Examples include medical imaging [15] and self-driving cars [5]. Unfortunately, several studies have shown that neural networks are not always robust to dataset shift [46, 26], nor do they exhibit calibrated predictive uncertainty, resulting in incorrect predictions made with high confidence [21].

Deep ensembles [33] achieve state-of-the-art results for predictive uncertainty calibration and robustness to dataset shift. Notably, they have been shown to outperform various approximate Bayesian neural networks [33, 46, 22]. Deep ensembles are constructed by training a *fixed* architecture multiple times with different random initializations. Due to the multi-modal loss landscape [18, 54], *randomization* by different initializations induces diversity among the base learners to yield a model with better uncertainty estimates than any of the individual base learners (i.e. ensemble members).

Our work focuses on *automatically* selecting *varying* base learner architectures in the ensemble, exploiting architectural variation as a beneficial source of diversity missing in deep ensembles due to their *fixed* architecture. Such architecture selection during ensemble construction allows a more "ensemble-aware" choice of architectures and is based on data rather than manual biases. As

[*]Equal contribution.

35th Conference on Neural Information Processing Systems (NeurIPS 2021), Sydney, Australia.

discussed in Section 2, while ensembles with varying architectures has already been explored in the literature, variation in architectures is typically limited to just varying depth and/or width, in contrast to more complex variations, such as changes in the topology of the connections and operations used, as considered in our work. More generally, automatic ensemble construction is well-explored in AutoML [17, 36, 45, 43]. Our work builds on this by demonstrating that, in the context of *uncertainty estimation*, automatically constructed ensembles with varying architectures outperform deep ensembles that use state-of-the-art, or even *optimal*, architectures (Figure 1). Studied under controlled settings, we assess the ensembles by various measures, including predictive performance, uncertainty estimation and calibration, base learner performance and two ensemble diversity metrics, showing that architectural variation is beneficial in ensembles.

Note that, *a priori*, it is not obvious how to find a set of diverse architectures that work well as an ensemble. On the one hand, optimizing the base learners' architectures in isolation may yield multiple base learners with similar architectures (like a deep ensemble). On the other hand, selecting the architectures randomly may yield numerous base learners with poor architectures harming the ensemble. Moreover, as in neural architecture search (NAS), we face the challenge of needing to traverse vast architectural search spaces. We address these challenges in the problem of *Neural Ensemble Search* (NES), an extension of NAS that aims to find a *set* of complementary architectures that together form a strong ensemble. In summary, our contributions are as follows:

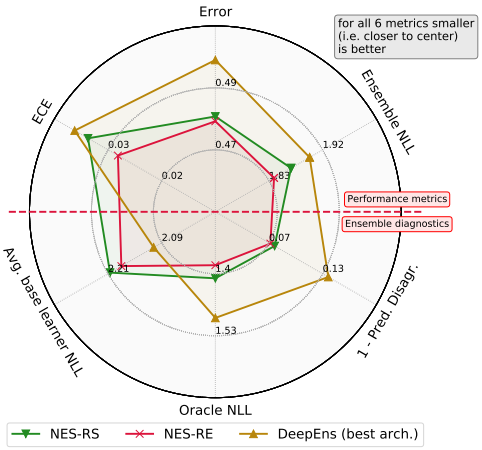


Figure 1: A comparison of a deep ensemble with the best architecture (out of 15,625 possible architectures) and ensembles constructed by our method NES on ImageNet-16-120 over NAS-Bench-201. Performance metrics (smaller is better): error, negative log-likelihood (NLL) and expected calibration error (ECE). We also measure average base learner NLL and two metrics for ensemble diversity (see Section 3): oracle NLL and [1−predictive disagreement]; small means more diversity for both metrics. NES ensembles outperform the deep ensemble, despite the latter having a significantly stronger average base learner.

1. We present two NES algorithms for automatically constructing ensembles with varying base learner architectures. As a first step, we present NES with random search (NES-RS), which is simple and easily parallelizable. We further propose NES-RE inspired by regularized evolution [48], which evolves a population of architectures yielding performant and robust ensembles.
2. This work is the first to apply automatic ensemble construction over architectures to *complex*, state-of-the-art neural architecture search spaces.
3. In the context of uncertainty estimation and robustness to dataset shift, we demonstrate that ensembles constructed by NES improve upon state-of-the-art deep ensembles. We validate our findings over five datasets and two architecture search spaces.

2 Related Work

Ensembles and uncertainty estimation. Ensembles of neural networks [23, 32, 11] are commonly used to boost performance. In practice, strategies for building ensembles include independently training multiple initializations of the same network, i.e. *deep ensembles* [33], training base learners on different bootstrap samples of the data [62], training with diversity-encouraging losses [40, 35, 60, 51, 29, 47] and using checkpoints during the training trajectory of a network [27, 41]. Despite a variety of approaches, Ashukha *et al.* [1] found many sophisticated ensembling techniques to be equivalent to a small-sized deep ensemble by test performance.

Much recent interest in ensembles has been due to their state-of-the-art predictive uncertainty estimates, with extensive empirical studies [46, 22] observing that deep ensembles outperform other approaches for uncertainty estimation, notably including Bayesian neural networks [4, 19, 52] and post-hoc calibration [21]. Although deep ensembles are not, technically speaking, equivalent to Bayesian neural networks and the relationship between the two is not well understood, diversity

among base learners in a deep ensemble yields a model which is arguably closer to *exact* Bayesian model averaging than other approximate Bayesian methods that only capture a single posterior mode in a multi-modal landscape [54, 18]. Also, He *et al.* [24] draw a rigorous link between Bayesian methods and deep ensembles for wide networks, and Pearce *et al.* [47] propose a technique for approximately Bayesian ensembling. Our primary baseline is deep ensembles as they provide state-of-the-art results in uncertainty estimation.

AutoML and ensembles of varying architectures. Automatic ensemble construction is commonly used in AutoML [17, 28]. Prior work includes use of Bayesian optimization to tune non-architectural hyperparameters of an ensemble's base learners [36], posthoc ensembling of fully-connected networks evaluated by Bayesian optimization [43] and building ensembles by iteratively adding (sub-)networks to improve ensemble performance [10, 42]. Various approaches, including ours, rely on ensemble selection [7]. We also note that Simonyan & Zisserman [49], He *et al.* [25] employ ensembles with varying architectures but *without* automatic construction. Importantly, in contrast to our work, all aforementioned works limit architectural variation to only changing width/depth or fully-connected networks. Moreover, such ensembles have not been considered before in terms of uncertainty estimation. Another important part of AutoML is neural architecture search (NAS), the process of automatically designing *single model* architectures [14], using strategies such as reinforcement learning [63], evolutionary algorithms [48] and gradient-based methods [39]. We use the search spaces defined by Liu *et al.* [39] and Dong & Yang [13], two of the most commonly used ones in recent literature.

Concurrent to our work, Wenzel *et al.* [53] consider ensembles with base learners having varying hyperparameters using an approach similar to NES-RS. However, they focus on non-architectural hyperparameters such as L_2 regularization strength and dropout rates, keeping the architecture fixed. As in our work, they also consider predictive uncertainty calibration and robustness to shift, finding similar improvements over deep ensembles.

3 Visualizing Ensembles of Varying Architectures

In this section, we discuss diversity in ensembles with varying architectures and visualize base learner predictions to add empirical evidence to the intuition that architectural variation results in more diversity. We also define two metrics for measuring diversity used later in Section 5.

3.1 Definitions and Set-up

Let $\mathcal{D}_{\text{train}} = \{(\boldsymbol{x}_i, y_i) : i = 1, \ldots, N\}$ be the training dataset, where the input $\boldsymbol{x}_i \in \mathbb{R}^D$ and, assuming a classification task, the output $y_i \in \{1, \ldots, C\}$. We use $\mathcal{D}_{\text{val}}$ and $\mathcal{D}_{\text{test}}$ for the validation and test datasets, respectively. Denote by f_θ a neural network with weights θ, so $f_\theta(\boldsymbol{x}) \in \mathbb{R}^C$ is the predicted probability vector over the classes for input $\boldsymbol{x}$. Let $\ell(f_\theta(\boldsymbol{x}), y)$ be the neural network's loss for data point $(\boldsymbol{x}, y)$. Given M networks $f_{\theta_1}, \ldots, f_{\theta_M}$, we construct the *ensemble* F of these networks by averaging the outputs, yielding $F(\boldsymbol{x}) = \frac{1}{M}\sum_{i=1}^{M} f_{\theta_i}(\boldsymbol{x})$.

In addition to the ensemble's loss $\ell(F(\boldsymbol{x}), y)$, we will also consider the *average base learner* loss and the *oracle ensemble's* loss. The average base learner loss is simply defined as $\frac{1}{M}\sum_{i=1}^{M} \ell(f_{\theta_i}(\boldsymbol{x}), y)$; we use this to measure the *average base learner strength* later. Similar to prior work [35, 60], the oracle ensemble F_{OE} composed of base learners $f_{\theta_1}, \ldots, f_{\theta_M}$ is defined to be the function which, given an input $\boldsymbol{x}$, returns the prediction of the base learner with the smallest loss for $(\boldsymbol{x}, y)$, that is,

$$F_{\text{OE}}(\boldsymbol{x}) = f_{\theta_k}(\boldsymbol{x}), \quad \text{where} \quad k \in \operatorname*{argmin}_i \ell(f_{\theta_i}(\boldsymbol{x}), y).$$

The oracle ensemble can only be constructed if the true class y is known. We use the oracle ensemble loss as one of the measures of *diversity* in base learner predictions. Intuitively, if base learners make diverse predictions for $\boldsymbol{x}$, the oracle ensemble is more likely to find some base learner with a small loss, whereas if all base learners make identical predictions, the oracle ensemble yields the same output as any (and all) base learners. Therefore, as a rule of thumb, all else being equal, smaller oracle ensemble loss indicates more diverse base learner predictions.

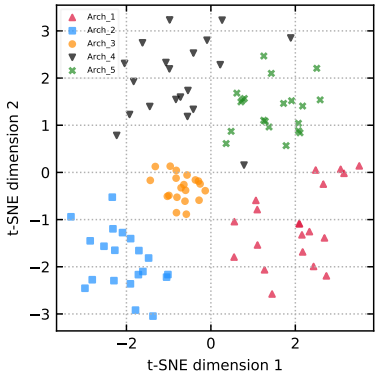
Proposition 3.1. Suppose ℓ is negative log-likelihood (NLL). Then, the oracle ensemble loss, ensemble loss, and average base learner loss satisfy the following inequality:

$$\ell(F_{\text{OE}}(\boldsymbol{x}), y) \leq \ell(F(\boldsymbol{x}), y) \leq \frac{1}{M}\sum_{i=1}^{M} \ell(f_{\theta_i}(\boldsymbol{x}), y).$$

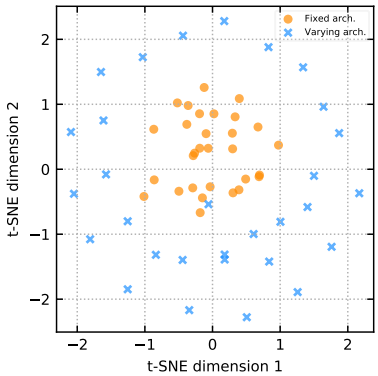
We refer to Appendix A for a proof. Proposition 3.1 suggests that it can be beneficial for ensembles to not only have strong average base learners (smaller upper bound), but also more diversity in their predictions (smaller lower bound). There is extensive theoretical work relating strong base learner performance and diversity with the generalization properties of ensembles [23, 61, 6, 30, 3, 20]. In Section 5, the two metrics we use for measuring diversity are oracle ensemble loss and (normalized) predictive disagreement, defined as the average pairwise predictive disagreement amongst the base learners, normalized by their average error [18].

3.2 Visualizing Similarity in Base Learner Predictions

The fixed architecture used to build deep ensembles is typically chosen to be a strong stand-alone architecture, either hand-crafted or found by NAS. However, optimizing the base learner's architecture and *then* constructing a deep ensemble can neglect diversity in favor of strong base learner performance. Having base learner architectures vary allows more diversity in their predictions. We provide empirical evidence for this intuition by visualizing the base learners' predictions. Fort *et al.* [18] found that base learners in a deep ensemble explore different parts of the function space by means of applying dimensionality reduction to their predictions. Building on this, we uniformly sample five architectures from the DARTS search space [39], train 20 initializations of each architecture on CIFAR-10 and visualize the similarity among the networks' predictions on the test dataset using t-SNE [50]. Experiment details are available in Section 5 and Appendix B.



(a) Five different architectures, each trained with 20 different initializations.



(b) Predictions of base learners in two ensembles, one with fixed architecture and one with varying architectures.

Figure 2: t-SNE visualization of base learner predictions. Each point corresponds to one network's (dimension-reduced) predictions.

As shown in Figure 2a, we observe clustering of predictions made by different initializations of a fixed architecture, suggesting that base learners with varying architectures explore different parts of the function space. Moreover, we also visualize the predictions of base learners of two ensembles, each of size $M = 30$, where one is a deep ensemble and the other has varying architectures (found by NES-RS as presented in Section 4). Figure 2b shows more diversity in the ensemble with varying architectures than in the deep ensemble. These qualitative findings can be quantified by measuring diversity: for the two ensembles shown in Figure 2b, we find the predictive disagreement to be 94.6% for the ensemble constructed by NES and 76.7% for the deep ensemble (this is consistent across independent runs). This indicates higher predictive diversity in the ensemble with varying architectures, in line with the t-SNE results.

4 Neural Ensemble Search

In this section, we define *neural ensemble search* (NES). In summary, a NES algorithm optimizes the architectures of base learners in an ensemble to minimize ensemble loss.

Given a network $f : \mathbb{R}^D \to \mathbb{R}^C$, let $\mathcal{L}(f, \mathcal{D}) = \sum_{(\boldsymbol{x},y)\in\mathcal{D}} \ell(f(\boldsymbol{x}), y)$ be the loss of f over dataset $\mathcal{D}$. Given a set of base learners $\{f_1, \ldots, f_M\}$, let `Ensemble` be the function which maps $\{f_1, \ldots, f_M\}$ to the ensemble $F = \frac{1}{M}\sum_{i=1}^{M} f_i$ as defined in Section 3 . To emphasize the architecture, we use the notation $f_{\theta,\alpha}$ to denote a network with architecture $\alpha \in \mathcal{A}$ and weights θ, where $\mathcal{A}$ is a search space (SS) of architectures. A NES algorithm aims to solve the following optimization problem:

$$\min_{\alpha_1,\ldots,\alpha_M\in\mathcal{A}} \mathcal{L}\left(\texttt{Ensemble}(f_{\theta_1,\alpha_1}, \ldots, f_{\theta_M,\alpha_M}), \mathcal{D}_{\text{val}}\right) \tag{1}$$
$$\text{s.t.} \quad \theta_i \in \operatorname*{argmin}_{\theta} \mathcal{L}(f_{\theta,\alpha_i}, \mathcal{D}_{\text{train}}) \qquad \text{for } i = 1, \ldots, M$$

Eq. 1 is difficult to solve for at least two reasons. First, we are optimizing over M architectures, so the search space is effectively $\mathcal{A}^M$, compared to it being $\mathcal{A}$ in typical NAS, making it more difficult

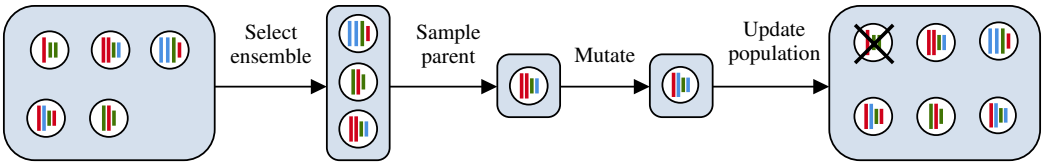


Figure 3: Illustration of one iteration of NES-RE. Network architectures are represented as colored bars of different lengths illustrating different layers and widths. Starting with the current population, ensemble selection is applied to select parent candidates, among which one is sampled as the parent. A mutated copy of the parent is added to the population, and the oldest member is removed.

to explore fully. Second, a larger search space also increases the risk of overfitting the ensemble loss to $\mathcal{D}_{\text{val}}$. A possible approach here is to consider the ensemble as a single large network to which we apply NAS, but joint training of an ensemble through a single loss has been empirically observed to underperform training base learners independently, specially for large networks [51]. Instead, our general approach to solve Eq. 1 consists of two steps:

1. **Pool building**: build a *pool* $\mathcal{P} = \{f_{\theta_1,\alpha_1}, \ldots, f_{\theta_K,\alpha_K}\}$ of size K consisting of potential base learners, where each f_{θ_i,α_i} is a network trained independently on $\mathcal{D}_{\text{train}}$.
2. **Ensemble selection**: select M base learners (without replacement as discussed below) $f_{\theta_1^*,\alpha_1^*}, \ldots, f_{\theta_M^*,\alpha_M^*}$ from $\mathcal{P}$ to form an ensemble which minimizes loss on $\mathcal{D}_{\text{val}}$. (We set $K \geq M$.)

Step 1 reduces the options for the base learner architectures, with the intention to make the search more feasible and focus on strong architectures; step 2 then selects a performant ensemble. This procedure also ensures that the ensemble's base learners are trained independently. We use ensemble selection without replacement [7] for step 2. More specifically, this is forward step-wise selection; that is, given the set of networks $\mathcal{P}$, we start with an empty ensemble and add the network from $\mathcal{P}$ which minimizes ensemble loss on $\mathcal{D}_{\text{val}}$. We repeat this without replacement until the ensemble is of size M. `ForwardSelect`$(\mathcal{P}, \mathcal{D}_{\text{val}}, M)$ denotes the resulting set of M base learners selected from $\mathcal{P}$.

Note that selecting the ensemble from $\mathcal{P}$ is a combinatorial optimization problem; a greedy approach such as `ForwardSelect` is nevertheless effective as shown by Caruana *et al.* [7], while keeping computational overhead low, given the predictions of the networks on $\mathcal{D}_{\text{val}}$. We also experimented with various other ensemble selection algorithms, including weighted averaging, as discussed in Section 5.2 and Appendix C.8, finding `ForwardSelect` to perform best.

We have not yet discussed the algorithm for building the pool in step 1; we present two approaches, NES-RS (Section 4.1) and NES-RE (Section 4.2). NES-RS is a simple random search based algorithm, while NES-RE is based on regularized evolution [48], a state-of-the-art NAS algorithm. Note that while gradient-based NAS methods have recently become popular, they are not naively applicable in our setting as the base learner selection component `ForwardSelect` is typically non-differentiable.

4.1 NES with Random Search

In NAS, random search (RS) is a competitive baseline on carefully designed architecture search spaces [37, 57, 58]. Motivated by its success and simplicity, NES with random search (NES-RS) builds the pool $\mathcal{P}$ by independently sampling architectures uniformly with replacement from the search space $\mathcal{A}$ (and training them). Since the architectures of networks in $\mathcal{P}$ vary, applying ensemble selection is a simple way to exploit diversity, yielding a performant ensemble. Importantly, NES-RS is easy to parallelize, exactly like deep ensembles. See Algorithm 2 in Appendix B.3 for pseudocode.

4.2 NES with Regularized Evolution

A more guided approach for building the pool $\mathcal{P}$ is using regularized evolution (RE) [48]. While RS has the benefit of simplicity by sampling architectures uniformly, the resulting pool might contain many weak architectures, leaving few strong architectures for `ForwardSelect` to choose between. RE is an evolutionary algorithm used on NAS spaces. It explores the search space by evolving (via mutations) a *population* of architectures. We first briefly describe RE as background before NES-RE. RE starts with a randomly initialized fixed-size population of architectures. At each iteration, a subset of size m of the population is sampled, from which the best network by validation loss is selected as the parent. A mutated copy (e.g. changing an operation in the network, see Appendix

Algorithm 1: NES with Regularized Evolution

Data: Search space $\mathcal{A}$; ensemble size M; comp. budget K; $\mathcal{D}_{\text{train}}, \mathcal{D}_{\text{val}}$; population size P; number of parent candidates m.

1 Sample P architectures $\alpha_1, \ldots, \alpha_P$ independently and uniformly from $\mathcal{A}$.
2 Train each architecture α_i using $\mathcal{D}_{\text{train}}$, and initialize $\mathfrak{p} = \mathcal{P} = \{f_{\theta_1,\alpha_1}, \ldots, f_{\theta_P,\alpha_P}\}$.
3 **while** $|\mathcal{P}| < K$ **do**
4 Select m parent candidates $\{f_{\widetilde{\theta}_1,\widetilde{\alpha}_1}, \ldots, f_{\widetilde{\theta}_m,\widetilde{\alpha}_m}\} = \texttt{ForwardSelect}(\mathfrak{p}, \mathcal{D}_{\text{val}}, m)$.
5 Sample uniformly a parent architecture α from $\{\widetilde{\alpha}_1, \ldots, \widetilde{\alpha}_m\}$. `// $\alpha$ stays in $\mathfrak{p}$.`
6 Apply mutation to α, yielding child architecture β.
7 Train β using $\mathcal{D}_{\text{train}}$ and add the trained network $f_{\theta,\beta}$ to $\mathfrak{p}$ and $\mathcal{P}$.
8 Remove the oldest member in $\mathfrak{p}$. `// as done in RE [48].`
9 Select base learners $\{f_{\theta_1^*,\alpha_1^*}, \ldots, f_{\theta_M^*,\alpha_M^*}\} = \texttt{ForwardSelect}(\mathcal{P}, \mathcal{D}_{\text{val}}, M)$ by forward step-wise selection without replacement.
10 **return** ensemble $\texttt{Ensemble}(f_{\theta_1^*,\alpha_1^*}, \ldots, f_{\theta_M^*,\alpha_M^*})$

B.4 for examples of mutations) of the parent architecture, called the child, is trained and added to the population, and the oldest member of the population is removed, preserving the population size. This is iterated until the computational budget is reached, returning the *history*, i.e. all the networks evaluated during the search, from which the best model is chosen by validation loss.

Building on RE for NAS, we propose NES-RE to build the pool of potential base learners. NES-RE starts by randomly initializing a population $\mathfrak{p}$ of size P. At each iteration, we first apply `ForwardSelect` to the population to select an ensemble of size m, then we uniformly sample one base learner from this ensemble to be the parent. A mutated copy of the parent is added to $\mathfrak{p}$ and the oldest network is removed, as in usual regularized evolution. This process is repeated until the computational budget is reached, and the history is returned as the pool $\mathcal{P}$. See Algorithm 1 for pseudocode and Figure 3 for an illustration.

Also, note the distinction between the *population* and the *pool* in NES-RE: the population is evolved, whereas the pool is the set of all networks evaluated during evolution (i.e., the history) and is used post-hoc for selecting the ensemble. Moreover, `ForwardSelect` is used both for selecting m parent candidates (line 4 in NES-RE) and choosing the final ensemble of size M (line 9 in NES-RE). In general, $m \neq M$.

4.3 Ensemble Adaptation to Dataset Shift

Deep ensembles offer (some) robustness to dataset shift relative to training data. In general, one may not know the type of shift that occurs at test time. By using an ensemble, diversity in base learner predictions prevents the model from relying on one base learner's predictions which may not only be incorrect but also overconfident.

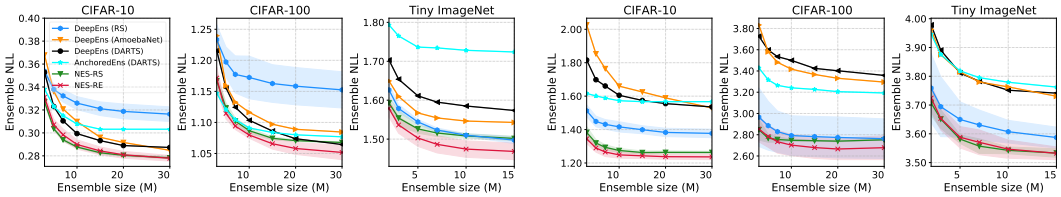
We assume that one does not have access to data points with test-time shift at training time, but one does have access to some validation data $\mathcal{D}_{\text{val}}^{\text{shift}}$ with a *validation* shift, which encapsulates one's belief about test-time shift. Crucially, test and validation shifts are disjoint. To adapt NES-RS and NES-RE to return ensembles robust to shift, we propose using $\mathcal{D}_{\text{val}}^{\text{shift}}$ instead of $\mathcal{D}_{\text{val}}$ whenever applying `ForwardSelect` to select the final ensemble. In Algorithms 1 and 2, this is in lines 9 and 3, respectively. Our experiments show that this is highly effective against shift.

Note that in line 4 of Algorithm 1, we can also replace $\mathcal{D}_{\text{val}}$ with $\mathcal{D}_{\text{val}}^{\text{shift}}$ when expecting test-time shift; we simply sample one of $\mathcal{D}_{\text{val}}, \mathcal{D}_{\text{val}}^{\text{shift}}$ uniformly at each iteration, in order to promote exploration of architectures that work well both in-distribution and during shift and reduce cost by avoiding running NES-RE once for each of $\mathcal{D}_{\text{val}}, \mathcal{D}_{\text{val}}^{\text{shift}}$. See Appendices C.1.3 and B.4 for further discussion.

5 Experiments

5.1 Comparison to Baselines: Uncertainty Estimation & Robustness to Dataset Shift

We compare NES to deep ensembles on different choices of architecture search space (DARTS [39] and NAS-Bench-201 [13] search spaces) and dataset (Fashion-MNIST, CIFAR-10, CIFAR-100,



(a) No data shift

(b) Dataset shift: severity 5 (out of 5)

Figure 4: NLL vs. ensemble sizes on CIFAR-10, CIFAR-100 and Tiny ImageNet with and without dataset shifts [26] over the DARTS search space. Mean NLL shown with 95% confidence intervals.

ImageNet-16-120 and Tiny ImageNet). The search spaces are *cell-based*, containing a rich variety of convolutional architectures with differing cell topologies, number of connections and operations (see Appendix B.1 for visualizations). For CIFAR-10/100 and Tiny ImageNet, we also consider dataset shifts proposed by Hendrycks & Dietterich [26]. We use NLL, classification error and expected calibration error (ECE) [21, 44] as our metrics, for which small values are better. Note that NLL and ECE evaluate predictive uncertainty. Experimental/implementation details are in Appendix B, additional experiments are in Appendix C. All evaluations are on the test dataset. Each paragraph below highlights one of our main findings.

Baselines. One of our main baselines is deep ensembles with fixed, optimized architectures. We consider various optimized architectures, indicated as "DeepEns (`arch`)":

- On the DARTS search space, we consider the architectures found by:
  1. the DARTS algorithm (DeepEns (`DARTS`)),
  2. regularized evolution (DeepEns (`AmoebaNet`)),
  3. random search, *with the same number of networks trained* as NES algorithms (DeepEns (`RS`)).
- On the NAS-Bench-201 search space, in addition to DeepEns (`RS`), we consider:
  1. the architecture found by GDAS[2] [12] (DeepEns (`GDAS`)),
  2. the best architecture in the search space by validation loss (DeepEns (`best arch.`)).[3]

We also compare to *anchored ensembles* [47], a recent technique for approximately Bayesian ensembles, which explicitly regularizes each base learner towards a fresh initialization sample and aims to induce more ensemble diversity. We use the DARTS architecture, and our implementation is described in Appendix B.5.

NES shows improved predictive uncertainty (NLL) and robustness to dataset shift (Figure 4). Figure 4a shows the NLL achieved by NES-RS, NES-RE and the baselines as functions of the ensemble size M without dataset shift. We find that NES algorithms consistently outperform deep ensembles, with NES-RE usually outperforming NES-RS. Next, we evaluate the robustness of the ensembles to dataset shift in Figure 4b. In our setup, all base learners are trained on $\mathcal{D}_{\text{train}}$ without data augmentation of shifted examples. However, as explained in Section 4.3, we use a shifted validation dataset, $\mathcal{D}_{\text{val}}^{\text{shift}}$, and evaluate on a shifted test dataset, $\mathcal{D}_{\text{test}}^{\text{shift}}$. The types of shifts appearing in $\mathcal{D}_{\text{val}}^{\text{shift}}$ are disjoint from those in $\mathcal{D}_{\text{test}}^{\text{shift}}$. The severity of the shift varies between 1-5. We refer to Appendix B and Hendrycks & Dietterich [26] for details. The fixed architecture used in the baseline DeepEns (RS) is selected based on its loss over $\mathcal{D}_{\text{val}}^{\text{shift}}$, but the DARTS and AmoebaNet are architectures from the literature. As shown in Figure 4b, ensembles picked by NES-RS and NES-RE are significantly more robust to dataset shift than the baselines, highlighting the effectiveness of applying `ForwardSelect` with $\mathcal{D}_{\text{val}}^{\text{shift}}$. Unsurprisingly, AnchoredEns (DARTS) and DeepEns (DARTS/AmoebaNet) perform poorly compared to the other methods, as they are not optimized to deal with dataset shift here. The results of our experiments on Fashion-MNIST are in Appendix C.1.1. In line with the results in this section, both NES algorithms outperform deep ensembles with NES-RE performing best.

Better uncertainty calibration versus dataset shift and classification error (Figure 5, Table 1). We also assess the ensembles by error and ECE. In short, ECE measures the mismatch between the model's confidence and accuracy. Figure 5 shows the ECE achieved by the ensembles at

[2]We did not consider the DARTS algorithm on NAS-Bench-201, since it returns degenerate architectures with poor performance on this space [13]. Whereas, GDAS yields state-of-the-art performance on this space.

[3]This is feasible, because all architectures in this search space were evaluated and are available.

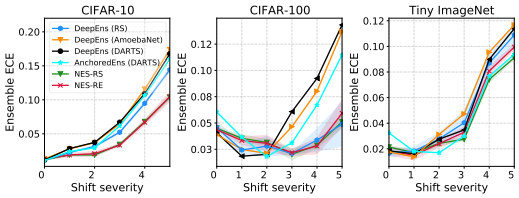


Figure 5: ECE vs. dataset shift severity on CIFAR-10, CIFAR-100 and Tiny ImageNet over the DARTS search space. No dataset shift is indicated as severity 0. Ensemble size is $M = 10$.

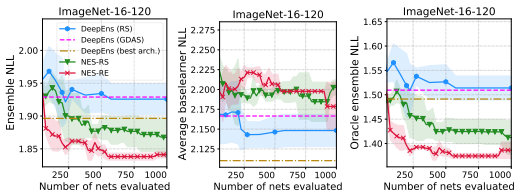


Figure 6: Ensemble, average base learner and oracle ensemble NLL versus budget K on ImageNet-16-120 over the NAS-Bench-201 search space. Ensemble size is $M = 3$.

varying dataset shift severities, noting that uncertainty calibration is especially important when models are used during dataset shift. Ensembles found with NES tend to exhibit better uncertainty calibration and are either competitive with or outperform anchored and deep ensembles for most shift severities. Notably, on CIFAR-10, ECE is reduced by up to 40% relative to the baselines. In terms of classification error, we find that ensembles constructed by NES outperform deep ensembles, with reductions of up to 7 percentage points in error, shown in Table 1. As with NLL, NES-RE tends to outperforms NES-RS.

Ensembles found by NES tend to be more diverse (Table 2). We measure ensemble diversity using the two metrics predictive disagreement (larger means more diverse) and oracle ensemble NLL (smaller means more diverse) as defined in Section 3 and average base learner NLL. Table 2 contrasts NES with the baselines in terms of these metrics. In terms of both diversity metrics, ensembles constructed by NES tend to be more diverse than anchored and deep ensembles. Ranking of the methods is largely consistent for different ensemble sizes M (see Appendix C). Unsurprisingly, note that the average base learner in NES is not always the best: by optimizing the fixed architecture first and *then* ensembling it, deep ensembles end up with a strong average base learner at the expense of less ensemble diversity. Despite higher average base learner NLL, NES ensembles perform better (recall Figure 4), highlighting once again the importance of diversity.

Table 1: Classification error of ensembles for different shift severities. Best values and all values within 95% confidence interval are bold faced. Note that NAS-Bench-201 comes with each architecture trained with *three* random initializations; therefore we set $M = 3$ in that case.

(a) $M = 10$; DARTS search space.

Dataset	Shift Severity	Classif. error (%), $\mathcal{A}$ = DARTS search space					
		DeepEns (RS)	DeepEns (Amoe.)	DeepEns (DARTS)	AnchoredEns (DARTS)	NES-RS	NES-RE
CIFAR-10	0	10.8±0.2	9.7	10.0	10.2	**9.4**±0.1	**9.4**±0.2
	3	25.1±0.7	25.6	26.3	28.6	23.2±0.2	**22.9**±0.2
	5	41.1±0.9	42.7	42.9	44.9	38.0±0.2	**37.4**±0.4
CIFAR-100	0	33.2±1.2	31.6	30.9	30.9	**30.7**±0.1	**30.4**±0.4
	3	54.8±1.6	54.2	55.1	55.2	**49.8**±0.1	**49.1**±1.0
	5	64.3±3.2	68.5	68.5	68.9	**62.4**±0.2	**61.4**±1.4
Tiny ImageNet	0	**37.5**±0.3	38.5	39.1	42.8	**37.4**±0.2	**37.0**±0.6
	3	53.8±0.6	55.4	54.6	58.7	53.0±0.2	**52.7**±0.2
	5	70.5±0.4	71.7	71.6	73.7	**69.9**±0.2	70.2±0.1

(b) $M = 3$; NAS-Bench-201 search space.

Dataset	Shift Severity	Classif. error (%), $\mathcal{A}$ = NAS-Bench-201 search space				
		DeepEns (GDAS)	DeepEns (best arch.)	DeepEns (RS)	NES-RS	NES-RE
CIFAR-10	0	8.4	**7.2**	7.8±0.2	7.7±0.1	7.6±0.1
	3	28.7	27.1	28.3±0.3	**22.0**±0.2	22.5±0.1
	5	47.8	46.3	37.1±0.0	**32.5**±0.2	33.0±0.5
CIFAR-100	0	29.9	26.4	26.3±0.4	**23.3**±0.3	**23.5**±0.2
	3	60.3	54.5	57.0±0.9	**46.6**±0.3	**46.7**±0.5
	5	75.3	69.9	64.5±0.0	**59.7**±0.2	60.0±0.6
ImageNet-16-120	0	49.9	49.9	50.5±0.6	**48.1**±1.0	**47.9**±0.4

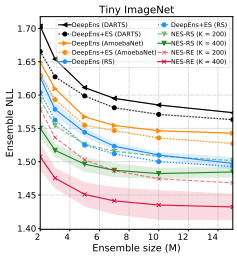
NES outperforms the deep ensemble of the best architecture in the NAS-Bench-201 search space (Figure 6). Next, we compare NES to deep ensembles over the NAS-Bench-201 search space, which has two benefits: we demonstrate that our findings are not specific to the DARTS search space, and NAS-Bench-201 is an exhaustively evaluated search space for which all architectures' trained weights are available (three initializations per architecture), allowing us to compare NES to the deep ensemble of the *best* architecture by validation loss. Results shown in Figure 6 compare the losses of the ensemble, average base learner and oracle ensemble versus the number of networks evaluated K. Interestingly, although DeepEns (best arch.) has a significantly stronger average base learner than the other methods, its lack of diversity, as indicated by higher oracle ensemble loss (Figure 6) and lower predictive disagreement (Figure 1), yields a weaker ensemble than both NES algorithms. Also, NES-RE outperforms NES-RS with a 6.6x speedup as shown in Figure 6-left.

5.2 Analysis and Ablations

Why does NES work? What if deep ensembles use ensemble selection over initializations? (Figure 7). NES algorithms differ from deep ensembles in two important ways: the ensembles use

Table 2: Diversity and base learner strength. Predictive disagreement (larger means more diverse) and oracle ensemble NLL (smaller means more diverse) are defined in Section 3. Despite the stronger base learners, deep ensembles tend to be less diverse than ensembles constructed by NES. The results are consistent across datasets and shift severities (Appendix C). Best values and all values within 95% confidence interval are bold faced.

Dataset	Metric	Method (with $M = 10$)					
		DeepEns (RS)	DeepEns (Amoe.)	DeepEns (DARTS)	AnchoredEns (DARTS)	NES-RS	NES-RE
CIFAR-10	Pred. Disagr.	$0.823_{\pm 0.023}$	**0.947**	0.932	0.842	$\mathbf{0.948}_{\pm 0.004}$	$0.943_{\pm 0.009}$
	Oracle NLL	$0.125_{\pm 0.007}$	0.103	0.093	0.113	$\mathbf{0.086}_{\pm 0.001}$	$0.088_{\pm 0.002}$
	Avg. bsl. NLL	$0.438_{\pm 0.005}$	0.552	0.513	**0.411**	$0.485_{\pm 0.003}$	$0.493_{\pm 0.010}$
CIFAR-100	Pred. Disagr.	$0.831_{\pm 0.027}$	**0.946**	0.935	0.839	$0.934_{\pm 0.006}$	$0.943_{\pm 0.014}$
	Oracle NLL	$0.635_{\pm 0.040}$	0.502	0.509	0.583	$\mathbf{0.498}_{\pm 0.008}$	$\mathbf{0.487}_{\pm 0.014}$
	Avg. bsl. NLL	$1.405_{\pm 0.028}$	1.552	1.491	**1.290**	$1.467_{\pm 0.022}$	$1.487_{\pm 0.036}$
Tiny ImageNet	Pred. Disagr.	$0.742_{\pm 0.008}$	0.737	0.749	0.662	$\mathbf{0.772}_{\pm 0.005}$	$\mathbf{0.768}_{\pm 0.005}$
	Oracle NLL	$0.956_{\pm 0.009}$	0.987	1.009	1.203	$0.929_{\pm 0.007}$	$\mathbf{0.910}_{\pm 0.015}$
	Avg. bsl. NLL	$1.743_{\pm 0.005}$	1.764	1.813	1.871	$1.755_{\pm 0.007}$	$\mathbf{1.728}_{\pm 0.012}$



Method	# nets trained	
	Arch.	Ensemble
DeepEns (DARTS)	32	10
DeepEns + ES (DARTS)	32	200
DeepEns (AmoebaNet)	25200	10
DeepEns + ES (AmoebaNet)	25200	200
DeepEns (RS)	200	10
DeepEns + ES (RS)	200	200
NES-RS ($K = 200$)	200	
NES-RS ($K = 400$)	400	
NES-RE ($K = 200$)	200	
NES-RE ($K = 400$)	400	

Figure 7: Comparison of NES to deep ensembles with ensemble selection on Tiny ImageNet. LEFT: NLL vs ensemble size. RIGHT: Cost for Tiny ImageNet experiments reported in terms of the number of networks trained when $M = 10$. The "arch" column indicates the number of networks trained to first select an architecture, and the "ensemble" column contains the number of networks trained to build the ensemble.

varying architectures and NES utilizes ensemble selection (`ForwardSelect`) to pick the base learners. On Tiny ImageNet over the DARTS search space, we conduct an experiment to explore whether the improvement offered by NES over deep ensembles is only due to ensemble selection. The baselines "DeepEns + ES" operate as follows: we optimize a fixed architecture for the base learners, train K random initializations of it to form a pool and apply `ForwardSelect` to select an ensemble of size M. Figure 7-left shows that both NES algorithms (each shown for two computational budgets $K = 200, 400$) outperform all DeepEns + ES baselines. Figure 7-right contains the cost of each method in terms of the number of networks trained. Note that DeepEns + ES (RS) is the most competitive of the deep ensemble baselines, and, at an equal budget of 400, it is outperformed by both NES algorithms. However, even at half the cost (200), NES-RE outperforms DeepEns + ES (RS) while NES-RS performs competitively. As expected, deep ensembles *with* ensemble selection consistently perform better than *without* ensemble selection at the expense of higher computational cost, but do not close the gap with NES algorithms. We also re-emphasize that comparisons between NES and deep ensembles in our experiments always fix the base learner training routine and method for composing the ensemble, so variations in ensemble performance are only due to the architecture choices. Therefore, to summarize, combined with the finding that NES outperforms deep ensembles even when NES' average base learner is weaker (as in e.g. Figure 6 and Table 2), we find architectural variation in ensembles to be important for NES' performance gains.

Computational cost and parallelizability. The primary computational cost involved in NES is training K networks to build the pool $\mathcal{P}$. In our experiments on the DARTS search space, we set K to be 400 for CIFAR-10/100 and 200 for Tiny ImageNet (except Figure 7 which additionally considers $K = 400$). Figure 7-right gives an example of costs for each method. The cost of a deep ensemble with an optimized architecture stems from the initial architecture search and the subsequent training of M random initializations to build ensemble. We refer to Appendix B for details and discussion of computation cost, including training times. We note that apart from DeepEns (DARTS) and AnchoredEns (DARTS), NES has a lower computational cost than the baselines in our experiments. Similar to deep ensembles, training the pool for NES-RS is embarrassingly parallel. Our implementation of NES-RE is also parallelized as described in Appendix B.

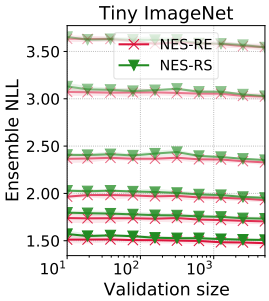
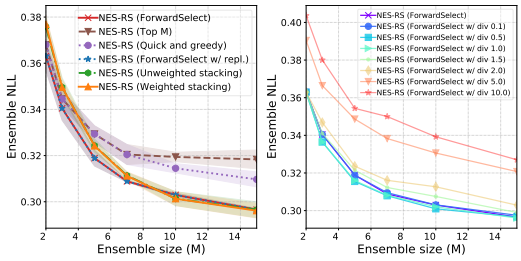


Figure 8: Test performance of NES algorithms with varying validation data sizes. Each curve corresponds to one particular dataset shift severity (0-5 with 0 being no shift). The more transparent curves correspond to higher shift severities.

Comparison of different ensemble selection algorithms. Both NES algorithms utilize `ForwardSelect` as the ensemble selection algorithm (ESA). We experimented with various other choices of ESAs, including weighted averaging and explicit diversity regularization, as shown in Figure 9. A detailed description of each ESA is in Appendix C.8. In summary, our choice of `ForwardSelect` performs better than or at par with all ESAs considered. Moreover, weighted averaging using stacking and Bayesian model averaging has



(a) Ensemble selection algorithms.

(b) Ensemble selection with explicit diversity regularization.

Figure 9: NES-RS on CIFAR-10.

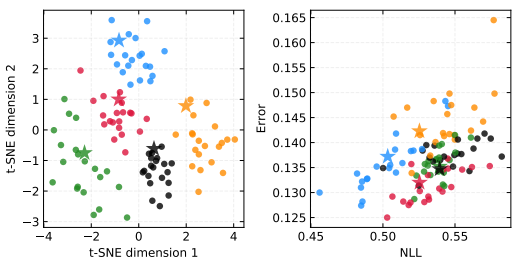


Figure 10: LEFT: t-SNE of the test predictions of 20 single random mutations (circles) from 5 parent architecture (stars). RIGHT: NLL and error achieved by these architectures.

a very minor impact since the weights end up being close to uniform. Explicit diversity regularization, as shown in Figure 9b, appears to slightly improve performance in some cases, provided that the diversity regularization strength hyperparameter is appropriately tuned.

NES is insensitive to the size of validation dataset $\mathcal{D}_{\text{val}}$. We study the sensitivity of NES to the size of $\mathcal{D}_{\text{val}}$. Specifically, we measure test loss of the ensembles selected using $\mathcal{D}_{\text{val}}$ of different sizes (with as few as 10 validation samples). The results in Figure 8 indicate that NES is insensitive to validation size for different levels of dataset shift severity, achieving almost the same loss when using $10\times$ fewer validation data. We provide more details in Appendix C.5 and we also discuss why overfitting to $\mathcal{D}_{\text{val}}$ is averted during ensemble selection.

Mutations of an architecture are similar in function space and by performance. To explore how NES-RE traverses the search space, we analyzed how mutations affect an architecture as follows. We sampled five random architectures from the DARTS space, and for each one, we applied a single random mutation twenty times, yielding a "family" of parent-children architectures, which are then trained on CIFAR-10. Figure 10-left shows the result of t-SNE applied to the test predictions of these architectures, where each color corresponds to one "family", of which the "star" is the parent architecture and the circles are the children architectures. The clustering demonstrates that architectures which differ by only a single mutation are similar in function space after training. Figure 10-right shows the NLL and error achieved by these architectures. Again, similar clustering shows that architectures differing by a single mutation also perform similarly w.r.t. NLL and error. This confirms that mutations allow for locally exploring the search space.

Further experiments. We also provide additional experiments in the Appendix. This includes a comparison of ensembles built by averaging logits vs. probabilities (Appendix C.7), a comparison of NES to ensembles with other hyperparameters being varied (either width/depth or training hyperparameters similar to Wenzel *et al.* [53]) (Appendix C.4) and using a weight-sharing model [2] as a proxy to accelerate the search phase in NES (Appendix C.9).

6 Conclusion, Limitations & Broader Impact

We presented Neural Ensemble Search for automatically constructing ensembles with varying architectures and demonstrated that the resulting ensembles outperform state-of-the-art deep ensembles in terms of uncertainty estimation and robustness to dataset shift. Our work highlights the benefit of ensembling varying architectures. In future work, we aim to address the limitation posed by the computational cost of NES due to building the pool $\mathcal{P}$. An interesting approach in this direction could be the use of differentiable NAS methods to simultaneously optimize base learner architectures within a one-shot model and reduce cost [39, 56, 8]. More generally, we also hope to explore what other hyperparameters can be varied to improve ensemble performance and how best to select them.

Our work can readily be applied to many existing ensemble-based deep learning systems to improve their predictive performance. This work also focuses on improving uncertainty estimation in neural networks, which is a key problem of growing importance with implications for safe deployment of such systems. We are not aware of any direct negative societal impacts of our work, since NES is task-agnostic and its impact depends on its applications.

Acknowledgments and Disclosure of Funding

AZ, TE and FH acknowledge support by the European Research Council (ERC) under the European Union Horizon 2020 research and innovation programme through grant no. 716721, and by BMBF grant DeToL. SZ acknowledges support from Aker Scholarship. CH wishes to acknowledge support from The Alan Turing Institute, The Medical Research Council UK, and the EPSRC Bayes4Health grant. The authors acknowledge support from ELLIS. We also thank Julien Siems for providing a parallel implementation of regularized evolution and Bobby He for useful comments on the manuscript.

References

[1] Arsenii Ashukha, Alexander Lyzhov, Dmitry Molchanov, and Dmitry Vetrov. Pitfalls of in-domain uncertainty estimation and ensembling in deep learning. In *International Conference on Learning Representations (ICLR)*, 2020.

[2] Gabriel Bender, Pieter-Jan Kindermans, Barret Zoph, Vijay Vasudevan, and Quoc Le. Understanding and simplifying one-shot architecture search. In *International Conference on Machine Learning*, 2018.

[3] Yijun Bian and Huanhuan Chen. When does Diversity Help Generalization in Classification Ensembles? *ArXiv*, abs/1903.06236, 2019.

[4] Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. Weight uncertainty in neural network. In *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 1613–1622, Lille, France, 07–09 Jul 2015. PMLR.

[5] Mariusz Bojarski, Davide Del Testa, Daniel Dworakowski, Bernhard Firner, Beat Flepp, Prasoon Goyal, Lawrence D. Jackel, Mathew Monfort, Urs Muller, Jiakai Zhang, Xin Zhang, Jake Zhao, and Karol Zieba. End to End Learning for Self-Driving Cars. *CoRR*, abs/1604.07316, 2016.

[6] Leo Breiman. Random forests. In *Machine Learning*, pages 5–32, 2001.

[7] Rich Caruana, Alexandru Niculescu-Mizil, Geoff Crew, and Alex Ksikes. Ensemble Selection from Libraries of Models. In *Proceedings of the Twenty-First International Conference on Machine Learning*, ICML '04, page 18, New York, NY, USA, 2004. Association for Computing Machinery.

[8] Xiangning Chen, Ruochen Wang, Minhao Cheng, Xiaocheng Tang, and Cho-Jui Hsieh. Dr{nas}: Dirichlet neural architecture search. In *International Conference on Learning Representations*, 2021.

[9] Patryk Chrabaszcz, Ilya Loshchilov, and Frank Hutter. A downsampled variant of imagenet as an alternative to the cifar datasets. *arXiv preprint arXiv:1707.08819*, 2017.

[10] Corinna Cortes, Xavier Gonzalvo, Vitaly Kuznetsov, Mehryar Mohri, and Scott Yang. AdaNet: Adaptive Structural Learning of Artificial Neural Networks. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 874–883, International Convention Centre, Sydney, Australia, 06–11 Aug 2017. PMLR.

[11] Thomas G. Dietterich. Ensemble Methods in Machine Learning. In *Multiple Classifier Systems*, pages 1–15, Berlin, Heidelberg, 2000. Springer Berlin Heidelberg.

[12] Xuanyi Dong and Yi Yang. Searching for a Robust Neural Architecture in Four GPU Hours. In *Computer Vision and Pattern Recognition (CVPR)*, pages 1761–1770, 2019.

[13] Xuanyi Dong and Yi Yang. Nas-bench-201: Extending the scope of reproducible neural architecture search. In *International Conference on Learning Representations (ICLR)*, 2020.

[14] Thomas Elsken, Jan Hendrik Metzen, and Frank Hutter. Neural Architecture Search: A Survey. *Journal of Machine Learning Research*, 20(55):1–21, 2019.

[15] Andre Esteva, Brett Kuprel, Roberto A. Novoa, Justin Ko, Susan M. Swetter, Helen M. Blau, and Sebastian Thrun. Dermatologist-level classification of skin cancer with deep neural networks. *Nature*, 542:115–, 2017.

[16] Stefan Falkner, Aaron Klein, and Frank Hutter. BOHB: Robust and Efficient Hyperparameter Optimization at Scale. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 1437–1446, Stockholmsmässan, Stockholm Sweden, 10–15 Jul 2018. PMLR.

[17] Matthias Feurer, Aaron Klein, Katharina Eggensperger, Jost Springenberg, Manuel Blum, and Frank Hutter. Efficient and robust automated machine learning. In *Advances in Neural Information Processing Systems 28*, pages 2962–2970. Curran Associates, Inc., 2015.

[18] Stanislav Fort, Huiyi Hu, and Balaji Lakshminarayanan. Deep Ensembles: A Loss Landscape Perspective. *ArXiv*, abs/1912.02757, 2019.

[19] Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *Proceedings of the 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 1050–1059, New York, New York, USA, 20–22 Jun 2016. PMLR.

[20] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016.

[21] Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q. Weinberger. On Calibration of Modern Neural Networks. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1321–1330, International Convention Centre, Sydney, Australia, 06–11 Aug 2017. PMLR.

[22] Fredrik K Gustafsson, Martin Danelljan, and Thomas B Schön. Evaluating Scalable Bayesian Deep Learning Methods for Robust Computer Vision. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2020.

[23] Lars K. Hansen and Peter Salamon. Neural network ensembles. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(10):993–1001, 1990.

[24] Bobby He, Balaji Lakshminarayanan, and Yee Whye Teh. Bayesian deep ensembles via the neural tangent kernel. In *Advances in Neural Information Processing Systems*, volume 33, pages 1010–1022. Curran Associates, Inc., 2020.

[25] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.

[26] Dan Hendrycks and Thomas Dietterich. Benchmarking Neural Network Robustness to Common Corruptions and Perturbations. In *International Conference on Learning Representations (ICLR)*, 2019.

[27] Gao Huang, Yixuan Li, Geoff Pleiss, Zhuang Liu, John Hopcroft, and Kilian Weinberger. Snapshot Ensembles: Train 1, get m for free. In *International Conference on Learning Representations (ICLR)*, 2017.

[28] F. Hutter, L. Kotthoff, and J. Vanschoren, editors. *Automated Machine Learning: Methods, Systems, Challenges*. Springer, 2018. In press, available at http://automl.org/book.

[29] Siddhartha Jain, Ge Liu, Jonas Mueller, and David Gifford. Maximizing overall diversity for improved uncertainty estimates in deep ensembles. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(04):4264–4271, Apr. 2020.

[30] Zhengshen Jiang, Hongzhi Liu, Bin Fu, and Zhonghai Wu. Generalized Ambiguity Decompositions for Classification with Applications in Active Learning and Unsupervised Ensemble Pruning. In *AAAI*, pages 2073–2079. AAAI Press, 2017.

[31] Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. Cifar-10 (canadian institute for advanced research), 2009.

[32] Anders Krogh and Jesper Vedelsby. Neural Network Ensembles, Cross Validation, and Active Learning. In *Advances in Neural Information Processing Systems 7*, pages 231–238. MIT Press, 1995.

[33] Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and Scalable Predictive Uncertainty Estimation using Deep Ensembles. In *Advances in Neural Information Processing Systems 30*, pages 6402–6413. Curran Associates, Inc., 2017.

[34] Y. Le and X. Yang. Tiny imagenet visual recognition challenge, 2015.

[35] Stefan Lee, Senthil Purushwalkam, Michael Cogswell, David Crandall, and Dhruv Batra. Why M Heads are Better than One: Training a Diverse Ensemble of Deep Networks. *arXiv e-prints*, page arXiv:1511.06314, 2015.

[36] Julien-Charles Lévesque, Christian Gagné, and Robert Sabourin. Bayesian hyperparameter optimization for ensemble learning. In *Proceedings of the Thirty-Second Conference on Uncertainty in Artificial Intelligence*, UAI'16, page 437–446, Arlington, Virginia, USA, 2016. AUAI Press.

[37] Liam Li and Ameet Talwalkar. Random Search and Reproducibility for Neural Architecture Search. In *UAI*, page 129. AUAI Press, 2019.

[38] Marius Lindauer and Frank Hutter. Best practices for scientific research on neural architecture search. *arXiv preprint arXiv:1909.02453*, 2019.

[39] Hanxiao Liu, Karen Simonyan, and Yiming Yang. DARTS: Differentiable Architecture Search. In *International Conference on Learning Representations (ICLR)*, 2019.

[40] Y. Liu and X. Yao. Ensemble learning via negative correlation. *Neural networks : the official journal of the International Neural Network Society*, 12 10:1399–1404, 1999.

[41] Ilya Loshchilov and Frank Hutter. SGDR: Stochastic Gradient Descent with Warm Restarts. In *International Conference on Learning Representations (ICLR)*, 2017.

[42] Vladimir Macko, Charles Weill, Hanna Mazzawi, and Javier Gonzalvo. Improving Neural Architecture Search Image Classifiers via Ensemble Learning. *ArXiv*, abs/1903.06236, 2019.

[43] Hector Mendoza, Aaron Klein, Matthias Feurer, Jost Tobias Springenberg, and Frank Hutter. Towards Automatically-Tuned Neural Networks. In *ICML 2016 AutoML Workshop*, 2016.

[44] Mahdi Pakdaman Naeini, Gregory F. Cooper, and Milos Hauskrecht. Obtaining Well Calibrated Probabilities using Bayesian Binning. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, AAAI'15, page 2901–2907. AAAI Press, 2015.

[45] Randal S. Olson and Jason H. Moore. Tpot: A tree-based pipeline optimization tool for automating machine learning. In *Workshop on Automatic Machine Learning*, volume 64 of *Proceedings of Machine Learning Research*, pages 66–74, New York, New York, USA, 24 Jun 2016. PMLR.

[46] Yaniv Ovadia, Emily Fertig, Jie Ren, Zachary Nado, D. Sculley, Sebastian Nowozin, Joshua Dillon, Balaji Lakshminarayanan, and Jasper Snoek. Can you trust your model's uncertainty? Evaluating predictive uncertainty under dataset shift. In *Advances in Neural Information Processing Systems 32*, pages 13991–14002. Curran Associates, Inc., 2019.

[47] Tim Pearce, Felix Leibfried, and Alexandra Brintrup. Uncertainty in neural networks: Approximately bayesian ensembling. In *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, volume 108 of *Proceedings of Machine Learning Research*, pages 234–244, Online, 26–28 Aug 2020. PMLR.

[48] Esteban Real, Alok Aggarwal, Yanping Huang, and Quoc V. Le. Regularized Evolution for Image Classifier Architecture Search. In *AAAI*, pages 4780–4789. AAAI Press, 2019.

[49] Karen Simonyan and Andrew Zisserman. Very Deep Convolutional Networks for Large-Scale Image Recognition. In *International Conference on Learning Representations (ICLR)*, 2015.

[50] Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9(2579-2605):85, 2008.

[51] Andrew M. Webb, Charles Reynolds, Dan-Andrei Iliescu, Henry W. J. Reeve, Mikel Luján, and Gavin Brown. Joint Training of Neural Network Ensembles. *CoRR*, abs/1902.04422, 2019.

[52] Max Welling and Yee Whye Teh. Bayesian learning via stochastic gradient langevin dynamics. In *Proceedings of the 28th International Conference on International Conference on Machine Learning*, ICML'11, page 681–688, Madison, WI, USA, 2011. Omnipress.

[53] Florian Wenzel, Jasper Snoek, Dustin Tran, and Rodolphe Jenatton. Hyperparameter ensembles for robustness and uncertainty quantification. In *Advances in Neural Information Processing Systems*, volume 33, pages 6514–6527. Curran Associates, Inc., 2020.

[54] Andrew G Wilson and Pavel Izmailov. Bayesian deep learning and a probabilistic perspective of generalization. In *Advances in Neural Information Processing Systems*, volume 33, pages 4697–4708. Curran Associates, Inc., 2020.

[55] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-MNIST: A Novel Image Dataset for Benchmarking Machine Learning Algorithms. *arXiv e-prints*, 2017.

[56] Yuhui Xu, Lingxi Xie, Xiaopeng Zhang, Xin Chen, Guo-Jun Qi, Qi Tian, and Hongkai Xiong. PC-DARTS: Partial Channel Connections for Memory-Efficient Architecture Search. In *International Conference on Learning Representations (ICLR)*, 2020.

[57] Antoine Yang, Pedro M. Esperança, and Fabio M. Carlucci. NAS evaluation is frustratingly hard. In *International Conference on Learning Representations (ICLR)*, 2020.

[58] Kaicheng Yu, Christian Sciuto, Martin Jaggi, Claudiu Musat, and Mathieu Salzmann. Evaluating the Search Phase of Neural Architecture Search. In *International Conference on Learning Representations (ICLR)*, 2020.

[59] Arber Zela, Julien Siems, and Frank Hutter. Nas-bench-1shot1: Benchmarking and dissecting one-shot neural architecture search. In *International Conference on Learning Representations*, 2020.

[60] Tianyi Zhou, Shengjie Wang, and Jeff A Bilmes. Diverse Ensemble Evolution: Curriculum Data-Model Marriage. In *Advances in Neural Information Processing Systems 31*, pages 5905–5916. Curran Associates, Inc., 2018.

[61] Zhi-Hua Zhou. *Ensemble Methods: Foundations and Algorithms*. Chapman & Hall, 1st edition, 2012.

[62] Zhi-Hua Zhou, Jianxin Wu, and Wei Tang. Ensembling neural networks: Many could be better than all. *Artificial Intelligence*, 137(1):239 – 263, 2002.

[63] Barret Zoph and Quoc V Le. Neural Architecture Search with Reinforcement Learning. In *International Conference on Learning Representations (ICLR)*, 2017.

[64] Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V Le. Learning Transferable Architectures for Scalable Image Recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8697–8710, 2018.

Supplementary Material for Neural Ensemble Search for Uncertainty Estimation and Dataset Shift

A Proof of Proposition 3.1

Taking the loss function to be NLL, we have $\ell(f(\boldsymbol{x}), y)) = -\log [f(\boldsymbol{x})]_y$, where $[f(\boldsymbol{x})]_y$ is the probability assigned by the network f of $\boldsymbol{x}$ belonging to the true class y, i.e. indexing the predicted probabilities $f(\boldsymbol{x})$ with the true target y. Note that $t \mapsto -\log t$ is a convex and decreasing function.

We first prove $\ell(F_{\text{OE}}(\boldsymbol{x}), y) \leq \ell(F(\boldsymbol{x}), y)$. Recall, by definition of F_{OE}, we have $F_{\text{OE}}(\boldsymbol{x}) = f_{\theta_k}(\boldsymbol{x})$ where $k \in \operatorname{argmin}_i \ell(f_{\theta_i}(\boldsymbol{x}), y)$, therefore $[F_{\text{OE}}(\boldsymbol{x})]_y = [f_{\theta_k}(\boldsymbol{x})]_y \geq [f_{\theta_i}(\boldsymbol{x})]_y$ for all $i = 1, \ldots, C$. That is, f_{θ_k} assigns the highest probability to the correct class y for input $\boldsymbol{x}$. Since $-\log$ is a decreasing function, we have

$$\begin{aligned}\ell(F(\boldsymbol{x}), y) &= -\log \left(\frac{1}{M} \sum_{i=1}^{M} [f_{\theta_i}(\boldsymbol{x})]_y \right) \\ &\geq -\log \left([f_{\theta_k}(\boldsymbol{x})]_y \right) = \ell(F_{\text{OE}}(\boldsymbol{x}), y).\end{aligned}$$

We apply Jensen's inequality in its finite form for the second inequality. Jensen's inequality states that for a real-valued, convex function φ with its domain being a subset of $\mathbb{R}$ and numbers $t_1, \ldots, t_n$ in its domain, $\varphi(\frac{1}{n} \sum_{i=1}^{n} t_i) \leq \frac{1}{n} \sum_{i=1}^{n} \varphi(t_i)$. Noting that $-\log$ is a convex function, $\ell(F(\boldsymbol{x}), y) \leq \frac{1}{M} \sum_{i=1}^{M} \ell(f_{\theta_i}(\boldsymbol{x}), y)$ follows directly.

B Experimental and Implementation Details

We describe details of the experiments shown in Section 5 and Appendix C and include Algorithms 2 and 3 describing NES-RS and `ForwardSelect`, respectively. Note that unless stated otherwise, all sampling over a discrete set is done uniformly in the discussion below.

Algorithm 2: NES with Random Search

Data: Search space $\mathcal{A}$; ensemble size M; comp. budget K; $\mathcal{D}_{\text{train}}, \mathcal{D}_{\text{val}}$.

Sample K architectures $\alpha_1, \ldots, \alpha_K$ independently and uniformly from $\mathcal{A}$.

Train each architecture α_i using $\mathcal{D}_{\text{train}}$, yielding a pool of networks $\mathcal{P} = \{f_{\theta_1,\alpha_1}, \ldots, f_{\theta_K,\alpha_K}\}$.

Select base learners $\{f_{\theta_1^*,\alpha_1^*}, \ldots, f_{\theta_M^*,\alpha_M^*}\} = \texttt{ForwardSelect}(\mathcal{P}, \mathcal{D}_{\text{val}}, M)$ by forward step-wise selection without replacement.

return ensemble $\texttt{Ensemble}(f_{\theta_1^*,\alpha_1^*}, \ldots, f_{\theta_M^*,\alpha_M^*})$

Algorithm 3: `ForwardSelect` (forward step-wise selection without replacement [7])

Data: Pool of base learners $\mathcal{P}$; ensemble size M; $\mathcal{D}_{\text{val}}$. Assume $|\mathcal{P}| \geq M$.

Initialize an empty set of base learners $E = \{\}$.

while $|E| < M$ **do**

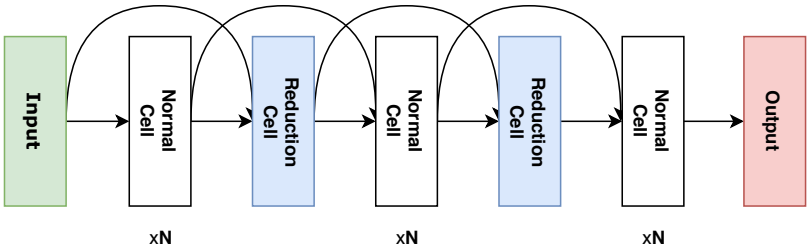
  Add $f_{\theta,\alpha}$ to E, where $f_{\theta,\alpha} \in \operatorname{argmin}_{f \in \mathcal{P}} \mathcal{L}(\texttt{Ensemble}(E \cup \{f\}), \mathcal{D}_{\text{val}})$

  Remove $f_{\theta,\alpha}$ from $\mathcal{P}$. `// without replacement.`

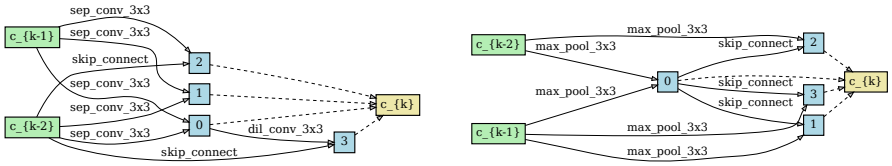
return E `// set of selected base learners.`

B.1 Architecture Search Spaces

DARTS search space. The first architecture search space we consider in our experiments is the one from DARTS [39]. We search for two types of *cells*: *normal* cells, which preserve the spatial dimensions, and *reduction* cells, which reduce the spatial dimensions. See Figure 11b for an illustration. The cells are stacked in a macro architecture, where they are repeated and connected using skip connections (shown in Figure 11a). Each cell is a directed acyclic graph, where nodes represent feature maps in the computational graph and edges between them correspond to operation choices (e.g. a convolution operation). The cell parses inputs from the previous and previous-previous cells in its 2 input nodes. Afterwards it contains 5 nodes: 4 intermediate nodes that aggregate the



(a) Macro architecture.



(b) Normal (left) and reduction (right) cells.

Figure 11: Illustration of the DARTS search space: (a) The macro architecture is a stack of normal and reduction cells. Normal cells are repeated N times between reduction cells ($N = 2$ in our experiments, i.e. 8 cells in total). (b) The cells for the DARTS architecture are depicted as directed acyclic graphs. Briefly, cells are composed of 2 input nodes (green), 4 intermediate nodes (blue) and one output node (red). Each edge is an operation which is applied to the preceding node's tensor and the output is summed element-wise in the succeeding node (an intermediate node). The output node is a concatenation of all intermediate nodes.

information coming from 2 previous nodes in the cell and finally an output node that concatenates the output of all intermediate nodes across the channel dimension. AmoebaNet contains one more intermediate node, making that a deeper architecture. The set of possible operations (eight in total in DARTS) that we use for each edge in the cells is the same as DARTS, but we leave out the "zero" operation since that is not necessary for non-differentiable approaches such as random search and evolution. Specifically, this leaves us with the following set of seven operations: 3×3 and 5×5 separable convolutions, 3×3 and 5×5 dilated separable convolutions, 3×3 average pooling, 3×3 max pooling and identity. Randomly of architectures is done by sampling the structure of the cell and the operations at each edge. The total number of architectures contained in this space is $\approx 10^{18}$. We refer the reader to Liu *et al.* [39] for more details.

NAS-Bench-201 search space. NAS-Bench-201 [13] is a tabular NAS benchmark, i.e. all architectures in the cell search space are trained and evaluated beforehand so one can query their performance (and weights) from a table quickly. Since this space is exhaustively evaluated, its size is also limited to only *normal* cells containing 4 nodes in total (1 input, 2 intermediate and 1 output node) and 5 operation choices on every edge connecting two nodes. This means that there are only 15,625 possible architecture configurations in this space. The networks are constructed by stacking 5 cells with in-between fixed residual blocks for reducing the spacial resolution. Each of them is trained for 200 epochs 3 times with 3 different seeds on 3 image classification datasets. For more details, please refer to Dong & Yang [13].

B.2 Datasets

Fashion-MNIST [55]. Fashion-MNIST consists of a training set of 60k 28×28 grayscale images and a test set of 10k images. The number of total labels is 10 classes. We split the 60k training set images to 50k used to train the networks and 10k used only for validation.

CIFAR-10/100 [31]. CIFAR-10 and CIFAR-100 both consist of 60k 32×32 colour images with 10 and 100 classes, respectively. We use 10k of the 60k training images as the validation set. We use the 10k original test set for final evaluation.

Tiny ImageNet [34]. Tiny Imagenet has 200 classes and each class has 500 training, 50 validation and 50 test colour images with 64×64 resolution. Since the original test labels are not available, we split the 10k validation examples into 5k for testing and 5k for validation.

ImageNet-16-120 [13] This variant of the ImageNet-16-120 [9] contains 151.7k train, 3k validation and 3k test ImageNet images downsampled to 16×16 and 120 classes.

Note that the test data points are only used for final evaluation. The data points for validation are used by the NES algorithms and DeepEns + ES baselines during ensemble selection and by DeepEns (RS) for picking the best architecture from the pool to use in the deep ensemble. Note that when considering dataset shift for CIFAR-10, CIFAR-100 and Tiny ImageNet, we also apply two disjoint sets of "corruptions" (following the terminology used by Hendrycks & Dietterich [26]) to the validation and test sets. We never apply any corruption to the training data. More specifically, out of the 19 different corruptions provided by Hendrycks & Dietterich [26], we randomly apply one from $\{$`Speckle Noise`, `Gaussian Blur`, `Spatter`, `Saturate`$\}$ to each data point in the validation set and one from $\{$`Gaussian Noise`, `Shot Noise`, `Impulse Noise`, `Defocus Blur`, `Glass Blur`, `Motion Blur`, `Zoom Blur`, `Snow`, `Frost`, `Fog`, `Brightness`, `Contrast`, `Elastic Transform`, `Pixelate`, `JPEG compression`$\}$ to each data point in the test set. This choice of validation and test corruptions follows the recommendation of Hendrycks & Dietterich [26]. Also, as mentioned in Section 5, each of these corruptions has 5 severity levels, which yields 5 corresponding severity levels for $\mathcal{D}_{\text{val}}^{\text{shift}}$ and $\mathcal{D}_{\text{test}}^{\text{shift}}$.

B.3 Training Routine & Time

The macro-architecture we use has 16 initial channels and 8 cells (6 normal and 2 reduction) and was trained using a batch size of 100 for 100 epochs for CIFAR-10/100 and 15 epochs for Fashion-MNIST. For Tiny ImageNet, we used 36 initial channels and a batch size of 128 for 100 epochs. Training a single network took roughly 40 minutes and 3 GPU hours[4] for CIFAR-10/100 and Tiny ImageNet, respectively. The networks are optimized using SGD with momentum set to 0.9. We used a learning rate of 0.025 for CIFAR-10/100 and Fashion-MNIST, and 0.1 for Tiny ImageNet. Unlike DARTS, we do not use any data augmentation procedure during training, nor any additional regularization such as ScheduledDropPath [64] or auxiliary heads, except for the case of Tiny ImageNet, for which we used ScheduledDropPath, gradient clipping and standard data augmentation as default in DARTS. All other hyperparameter settings are exactly as in DARTS [39].

All results containing error bars are averaged over multiple runs (at least five) with error bars indicating a 95% confidence interval. We used a budget $K = 400$ for CIFAR-10/100 (corresponding to 267 GPU hours). For Tiny ImageNet on the DARTS search space, unless otherwise stated, we used $K = 200$ (corresponding to 600 GPU hours). Note that only in Figure 7 and Table 30 we also tried NES algorithms using a higher budget of $K = 400$ on Tiny ImageNet for an equal-cost comparison with DeepEns + ES (RS). For ImageNet-16-120 on the NAS-Bench-201 search space, we used $K = 1000$ (compute costs are negligible since it is a tabular benchmark). Note that NES algorithms can be parallelized, especially NES-RS which is embarrassingly parallel like deep ensembles, therefore training time can be reduced easily when using multiple GPUs.

B.4 Implementation Details of NES-RE

Parallization. Running NES-RE on a single GPU requires evaluating hundreds of networks sequentially, which is tedious. To circumvent this, we distribute the "while $|\mathcal{P}| < K$" loop in Algorithm 1 over multiple GPUs, called worker nodes. We use the parallelism scheme provided by the `hpbandster` [16] codebase.[5] In brief, the master node keeps track of the population and history (lines 1, 4-6, 8 in Algorithm 1), and it distributes the training of the networks to the individual worker nodes (lines 2, 7 in Algorithm 1). In our experiments, we always use 20 worker nodes and evolve a population $\mathfrak{p}$ of size $P = 50$ when working over the DARTS search space. Over NAS-Bench-201, we used one worker since it is a tabular NAS benchmark and hence is quick to evaluate on. During iterations of evolution, we use an ensemble size of $m = 10$ to select parent candidates.

Mutations. We adapt the mutations used in RE to the DARTS search space. As in RE, we first pick a normal or reduction cell at random to mutate and then sample one of the following mutations:

- `identity`: no mutation is applied to the cell.
- `op mutation`: sample one edge in the cell and replace its operation with another operation sampled from the list of operations described in Appendix B.1.

[4] We used NVIDIA RTX 2080Ti GPUs for training.

[5] `https://github.com/automl/HpBandSter`

- `hidden state mutation`: sample one intermediate node in the cell, then sample one of its two incoming edges. Replace the input node of that edge with another sampled node, without altering the edge's operation.

For example, one possible mutation would be changing the operation on an edge in the cell (as shown in e.g. Figure 16) from say 5×5 separable convolution to 3×3 max pooling. See Real *et al.* [48] for further details and illustrations of these mutations. Note that for NAS-Bench-201, following Dong & Yang [13] we only use `op mutation`.

Adaptation of NES-RE to dataset shifts. As described in Section 4.3, at each iteration of evolution, the validation set used in line 4 of Algorithm 1 is sampled uniformly between $\mathcal{D}_{\text{val}}$ and $\mathcal{D}_{\text{val}}^{\text{shift}}$ when dealing with dataset shift. In this case, we use shift severity level 5 for $\mathcal{D}_{\text{val}}^{\text{shift}}$. Once the evolution is complete and the pool $\mathcal{P}$ has been formed, then for each severity level $s \in \{0, 1, \ldots, 5\}$, we apply `ForwardSelect` with $\mathcal{D}_{\text{val}}^{\text{shift}}$ of severity s to select an ensemble from $\mathcal{P}$ (line 9 in Algorithm 1), which is then evaluated on $\mathcal{D}_{\text{test}}^{\text{shift}}$ of severity s. (Here $s = 0$ corresponds to no shift.) This only applies to CIFAR-10, CIFAR-100 and Tiny ImageNet, as we do not consider dataset shift for Fashion-MNIST and ImageNet-16-120 .

B.5 Implementation details of anchored ensembles.

Anchored ensembles [47] are constructed by independently training M base learners, each regularized towards a new initialization sample (called the *anchor point*). Specifically, the k-th base learner f_{θ_k} is trained to minimize the following loss (using the notation in Section 3.1):

$$\frac{1}{N} \sum_{i=1}^{N} \ell(f_{\theta_k}(\boldsymbol{x}_i), y_i) + \frac{\lambda}{N} \|\Gamma^{1/2}(\theta_k - \mathring{\theta}_k)\|_2^2$$

where $\ell(f_{\theta_k}(\boldsymbol{x}_i), y_i) = -\log\left[f_{\theta_k}(\boldsymbol{x}_i)\right]_{y_i}$ is the cross-entropy loss for the i-th datapoint. Moreover, the anchor point $\mathring{\theta}_k$ is an independently sampled initialization, and λ is the regularization strength. Defining p to be the number of parameters, i.e. $\theta \in \mathbb{R}^p$, and letting σ_i^2 be the variance of the initialization for the i-th parameter $(\theta_k)_i$, $\Gamma \in \mathbb{R}^{p \times p}$ is the diagonal matrix with $\Gamma_{ii} = 1/(2\sigma_i^2)$.

Although Pearce *et al.* [47] do not include a tune-able regularization parameter λ (they set $\lambda = 1$), we found anchored ensembles performed poorly without tuning λ. For CIFAR-10/100, we used $\lambda = 0.4$ and for Tiny ImageNet, we used $\lambda = 0.1$ (tuned by grid search). We followed the recommendation of Pearce *et al.* [47] in using a different initialization for optimization than the anchor point $\mathring{\theta}_k$. Moreover, we turned off weight decay when training anchored ensembles, and we did not apply the regularization to the trainable parameters in batch normalization layers (which are initialized deterministically).

C Additional Experiments

In this section we provide additional results for the experiments conducted in Section 5. Note that, as with all results shown in Section 5, all evaluations are made on test data unless stated otherwise.

C.1 Supplementary plots complementing Section 5

C.1.1 Results on Fashion-MNIST

As shown in Figure 12, we see a similar trend on Fashion-MNIST as with other datasets: NES ensembles outperform deep ensembles with NES-RE outperforming NES-RS. To understand why NES algorithms outperform deep ensembles on Fashion-MNIST [55], we compare the average base learner loss (Figure 13) and oracle ensemble loss (Figure 14) of NES-RS, NES-RE and DeepEns (RS). Notice that, apart from the case when ensemble size $M = 30$, NES-RS and NES-RE find ensembles with both stronger and more diverse base learners (smaller losses in Figures 13 and 14, respectively). While it is expected that the oracle ensemble loss is smaller for NES-RS and NES-RE compared to DeepEns (RS), it initially appears surprising that DeepEns (RS) has a larger average base learner loss considering that the architecture for the deep ensemble is chosen to minimize the base learner loss. We found that this is due to the loss having a sensitive dependence not only on the architecture but also the initialization of the base learner networks. Therefore, re-training the best architecture by validation loss to build the deep ensemble yields base learners with higher losses due to the

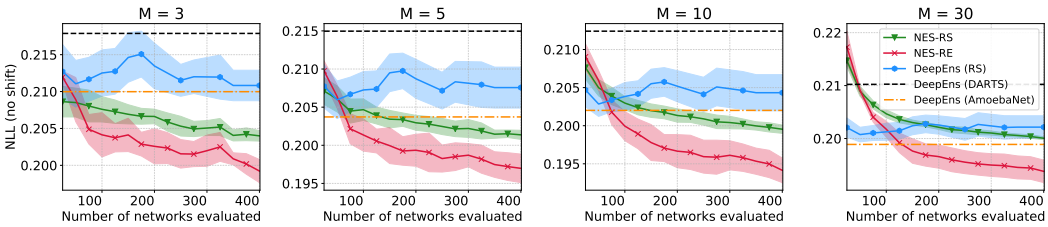


Figure 12: Results on Fashion-MNIST with varying ensembles sizes M. Lines show the mean NLL achieved by the ensembles with 95% confidence intervals.

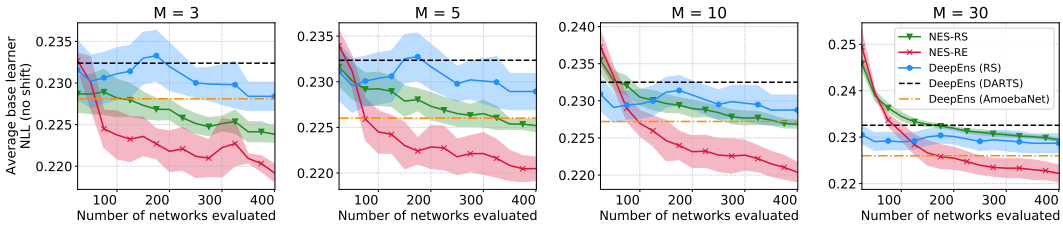


Figure 13: Average base learner loss for NES-RS, NES-RE and DeepEns (RS) on Fashion-MNIST. Lines show the mean NLL and 95% confidence intervals.

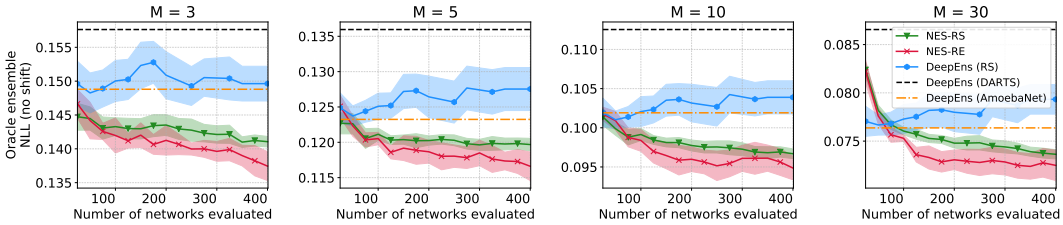


Figure 14: Oracle ensemble loss for NES-RS, NES-RE and DeepEns (RS) on Fashion-MNIST. Lines show the mean NLL and 95% confidence intervals.

use of different random initializations. Fortunately, NES algorithms are not affected by this, since they simply select the ensemble's base learners from the pool without having to re-train anything which allows them to exploit good architectures as well as initializations. Note that, for CIFAR-10-C experiments, this was not the case; base learner losses did not have as sensitive a dependence on the initialization as they did on the architecture.

In Table 3, we compare the classification error and expected calibration error (ECE) of NES algorithms with the deep ensembles baseline for various ensemble sizes on Fashion-MNIST. Similar to the loss, NES algorithms also achieve smaller errors, while ECE remains approximately the same for all methods.

C.1.2 Entropy on out-of-distribution inputs

To assess how well models respond to completely out-of-distribution (OOD) inputs (inputs which do not belong to one of the classes the model can predict), we investigate the entropy of the predicted probability distribution over the classes when the input is OOD. Higher entropy of the predicted probabilities indicates more uncertainty in the model's output. For CIFAR-10 on the DARTS search space, we compare the entropy of the predictions made by NES ensembles with deep ensembles on two types of OOD inputs: images from the SVHN dataset and Gaussian noise. In Figure 15, we notice that NES ensembles indicate higher uncertainty when given inputs of Gaussian noise than deep ensembles but behave similarly to deep ensembles for inputs from SVHN.

C.1.3 Additional Results on the DARTS search space

In this section, we provide additional experimental results on CIFAR-10, CIFAR-100 and Tiny ImageNet on the DARTS search space, complementing the results in Section 5 as shown in Figures 17-23. We also include examples of architectures chosen by NES-RE in Figure 16 for Tiny ImageNet, showcasing variation in architectures.

Table 3: Error and ECE of ensembles on Fashion-MNIST for different ensemble sizes M. Best values and all values within 95% confidence interval are bold faced.

M	**Classification Error** (out of 1)					**Expected Calibration Error (ECE)**				
	NES-RS	NES-RE	DeepEns (RS)	DeepEns (DARTS)	DeepEns (AmoebaNet)	NES-RS	NES-RE	DeepEns (RS)	DeepEns (DARTS)	DeepEns (AmoebaNet)
3	$0.074_{\pm 0.001}$	$\mathbf{0.072}_{\pm 0.001}$	$0.076_{\pm 0.001}$	0.077	0.077	$0.007_{\pm 0.001}$	$0.007_{\pm 0.002}$	$0.008_{\pm 0.001}$	**0.003**	0.008
5	$\mathbf{0.073}_{\pm 0.001}$	$\mathbf{0.071}_{\pm 0.002}$	$0.075_{\pm 0.001}$	0.077	0.074	$\mathbf{0.005}_{\pm 0.001}$	$\mathbf{0.005}_{\pm 0.001}$	$\mathbf{0.006}_{\pm 0.001}$	**0.005**	**0.005**
10	$0.073_{\pm 0.001}$	$\mathbf{0.070}_{\pm 0.001}$	$0.075_{\pm 0.001}$	0.076	0.073	$\mathbf{0.004}_{\pm 0.001}$	$\mathbf{0.005}_{\pm 0.001}$	$\mathbf{0.005}_{\pm 0.001}$	0.006	**0.005**
30	$0.073_{\pm 0.001}$	$\mathbf{0.070}_{\pm 0.001}$	$0.074_{\pm 0.001}$	0.075	0.073	$\mathbf{0.004}_{\pm 0.001}$	$\mathbf{0.004}_{\pm 0.002}$	$\mathbf{0.004}_{\pm 0.001}$	0.008	**0.004**

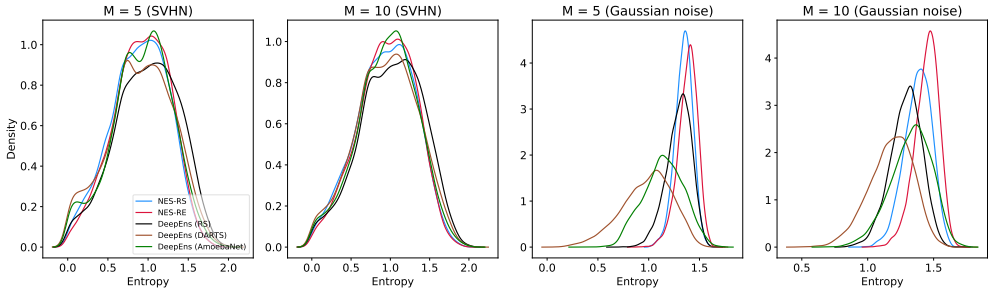
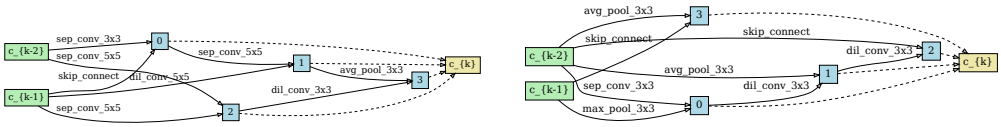
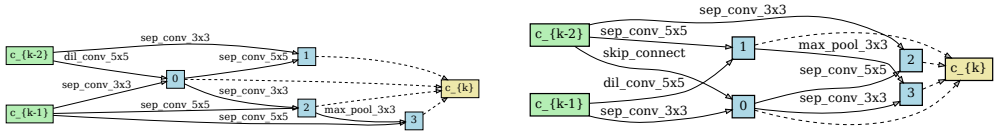


Figure 15: Entropy of predicted probabilities when trained on CIFAR-10 in the DARTS search space.

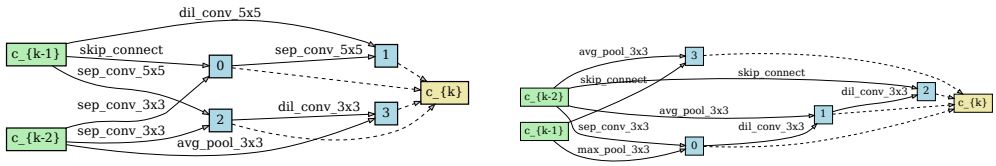
Results on CIFAR for larger models. In additional to the results on CIFAR-10 and CIFAR-100 on the DARTS search space using the settings described in Appendix B.3, we also train larger models (around 3M parameters) by scaling up the number of stacks cells and initial channels in the network. We run NES and other baselines similarly as done before and plot results in Figure 27 and 28 for NLL and classification test error with budget $K = 90$. As shown, NES algorithms tend to outperform or be competitive with the baselines. Note, more runs are needed including error bars for conclusive results in this case.



(a) First base learner

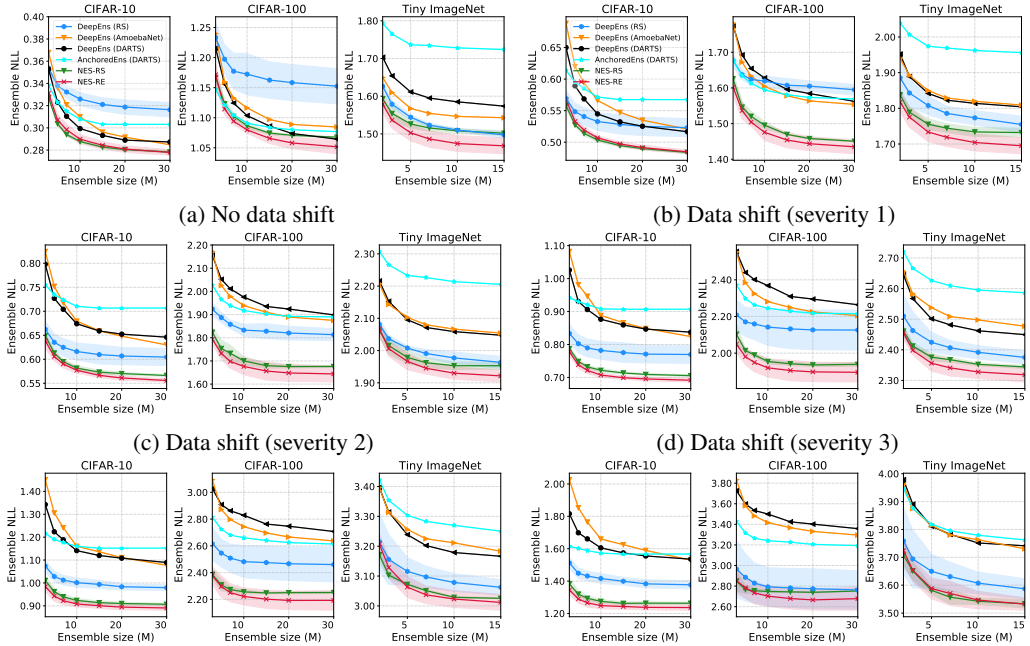


(b) Second base learner



(c) Third base learner

Figure 16: Illustration of example cells found by NES-RE for an ensemble of size $M = 3$ on Tiny ImageNet over the DARTS search space. In each sub-figure, left is the normal cell and right is the reduction cell. Note that the first and third base learners have the same reduction cell in this instance.

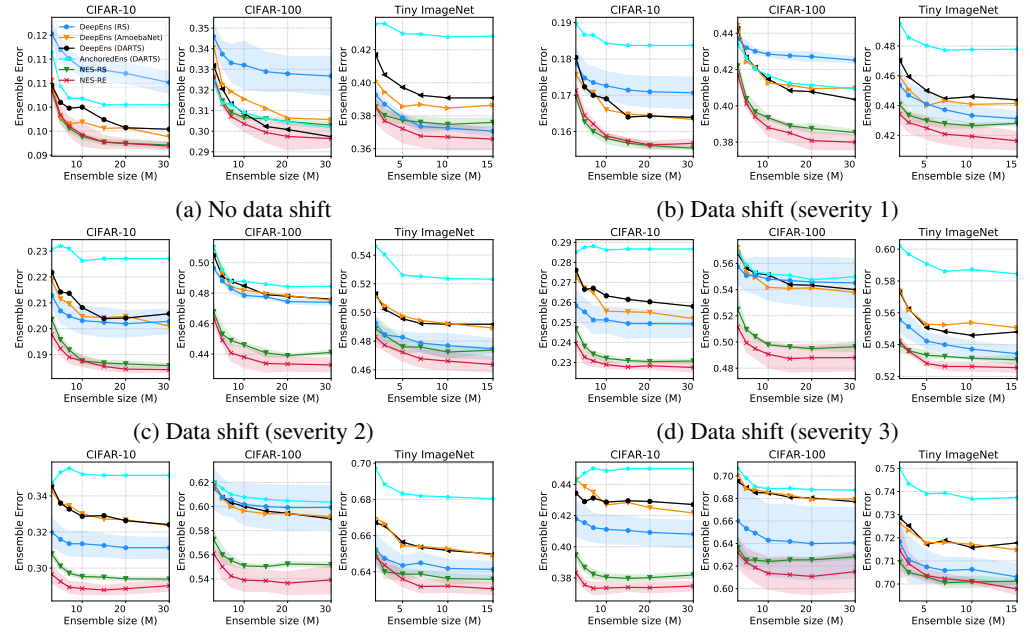


(a) No data shift (b) Data shift (severity 1)

(c) Data shift (severity 2) (d) Data shift (severity 3)

(e) Data shift (severity 4) (f) Data shift (severity 5)

Figure 17: NLL vs. ensemble sizes on CIFAR-10, CIFAR-100 and Tiny ImageNet with varying dataset shifts [26] over DARTS space.

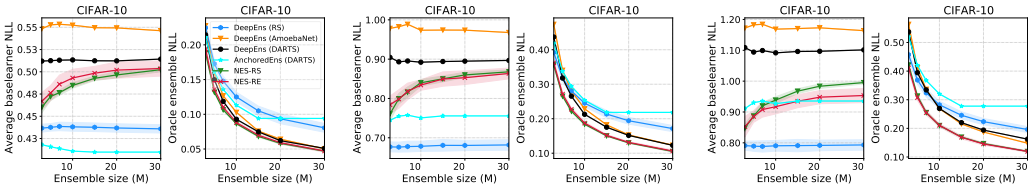


(a) No data shift (b) Data shift (severity 1)

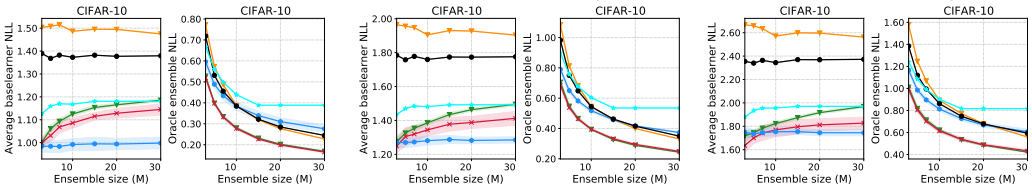
(c) Data shift (severity 2) (d) Data shift (severity 3)

(e) Data shift (severity 4) (f) Data shift (severity 5)

Figure 18: Classification error rate (between 0-1) vs. ensemble size on DARTS search space.

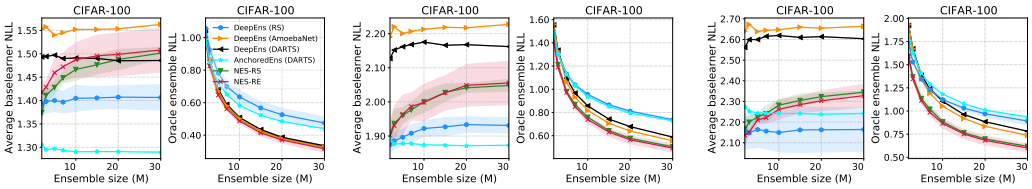


(a) No data shift (b) Data shift (severity 1) (c) Data shift (severity 2)

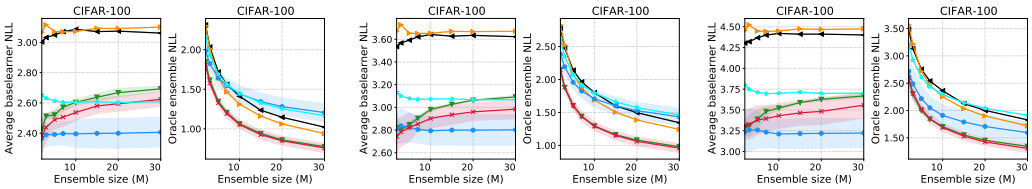


(d) Data shift (severity 3) (e) Data shift (severity 4) (f) Data shift (severity 5)

Figure 19: Average base learner and oracle ensemble NLL across ensemble sizes and shift severities on CIFAR-10 over DARTS search space.

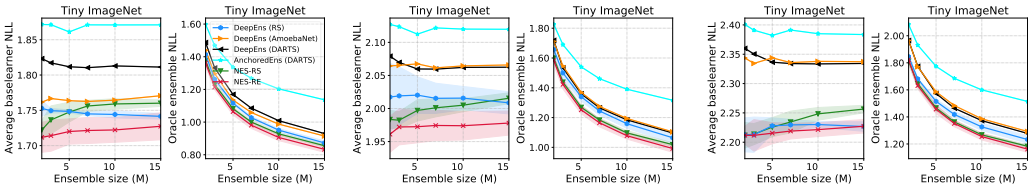


(a) No data shift (b) Data shift (severity 1) (c) Data shift (severity 2)

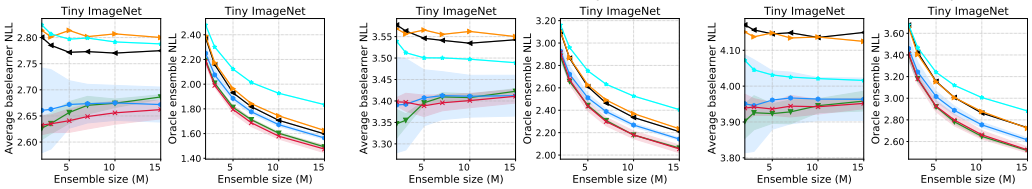


(d) Data shift (severity 3) (e) Data shift (severity 4) (f) Data shift (severity 5)

Figure 20: Average base learner and oracle ensemble NLL across ensemble sizes and shift severities on CIFAR-100 over DARTS search space.

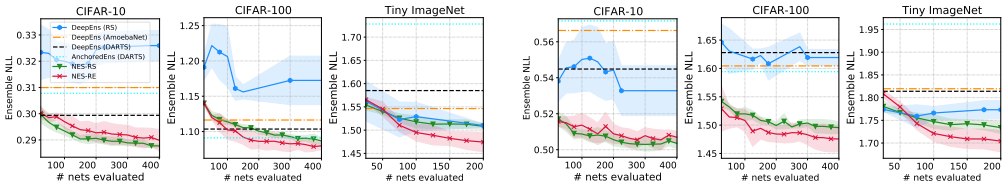


(a) No data shift (b) Data shift (severity 1) (c) Data shift (severity 2)

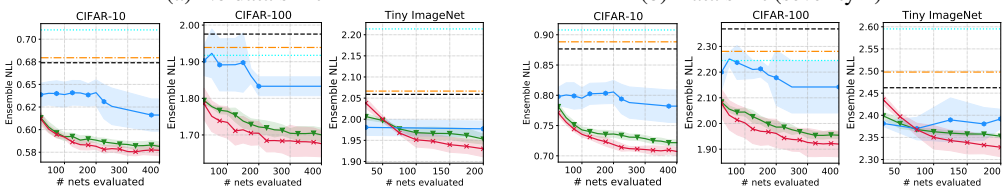


(d) Data shift (severity 3) (e) Data shift (severity 4) (f) Data shift (severity 5)

Figure 21: Average base learner and oracle ensemble NLL across ensemble sizes and shift severities on Tiny ImageNet over DARTS search space.



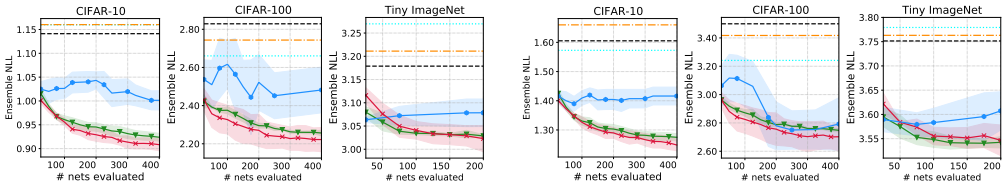
(a) No data shift



(b) Data shift (severity 1)

(c) Data shift (severity 2)

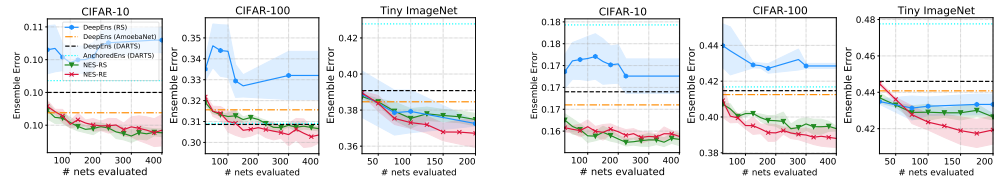
(d) Data shift (severity 3)



(e) Data shift (severity 4)

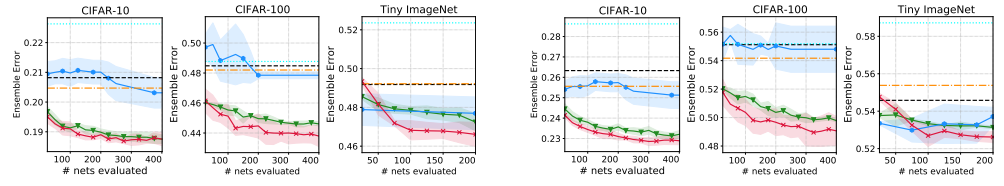
(f) Data shift (severity 5)

Figure 22: Ensemble NLL vs. budget K. Ensemble size fixed at $M = 10$.



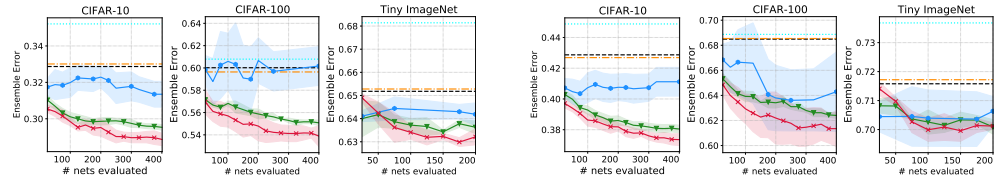
(a) No data shift

(b) Data shift (severity 1)



(c) Data shift (severity 2)

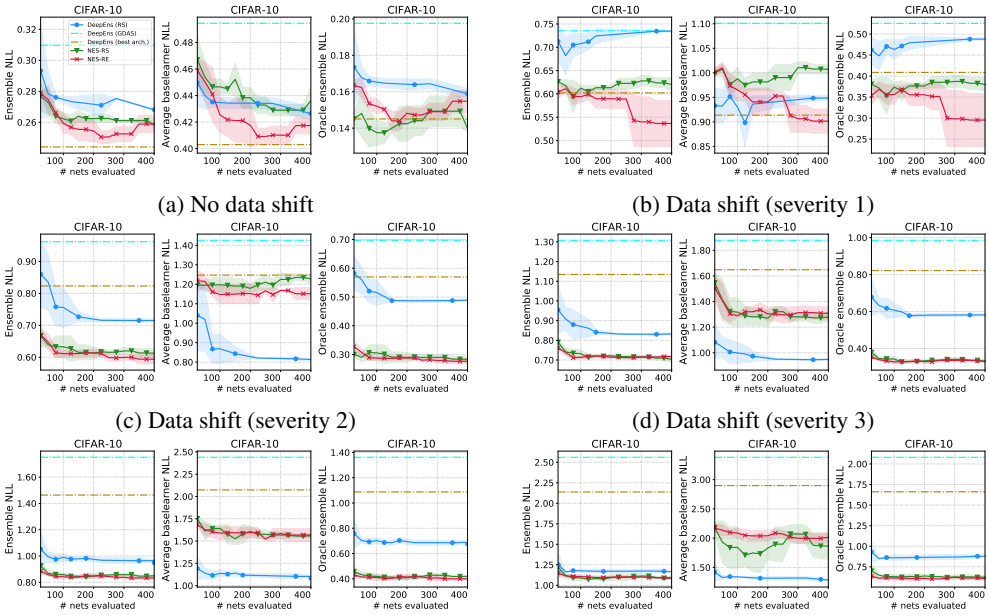
(d) Data shift (severity 3)



(e) Data shift (severity 4)

(f) Data shift (severity 5)

Figure 23: Ensemble error vs. budget K. Ensemble size fixed at $M = 10$.

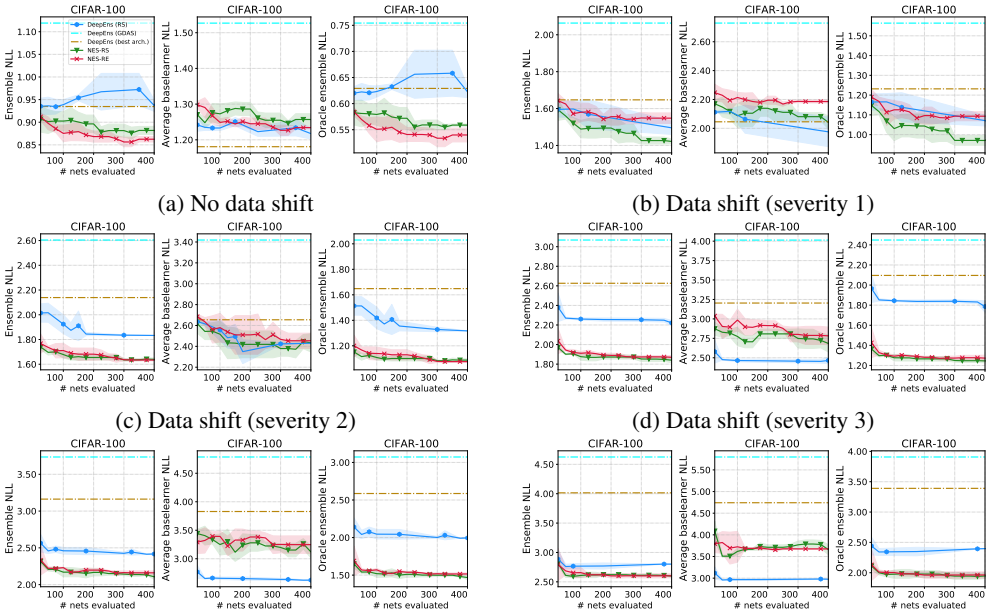


(a) No data shift (b) Data shift (severity 1)

(c) Data shift (severity 2) (d) Data shift (severity 3)

(e) Data shift (severity 4) (f) Data shift (severity 5)

Figure 24: NES and deep ensembles performance on the NAS-Bench-201 search space and CIFAR-10 dataset. Plots show ensemble NLL, average baselearner NLL and oracle ensemble NLL vs. budget K. Ensemble size fixed at $M = 3$.

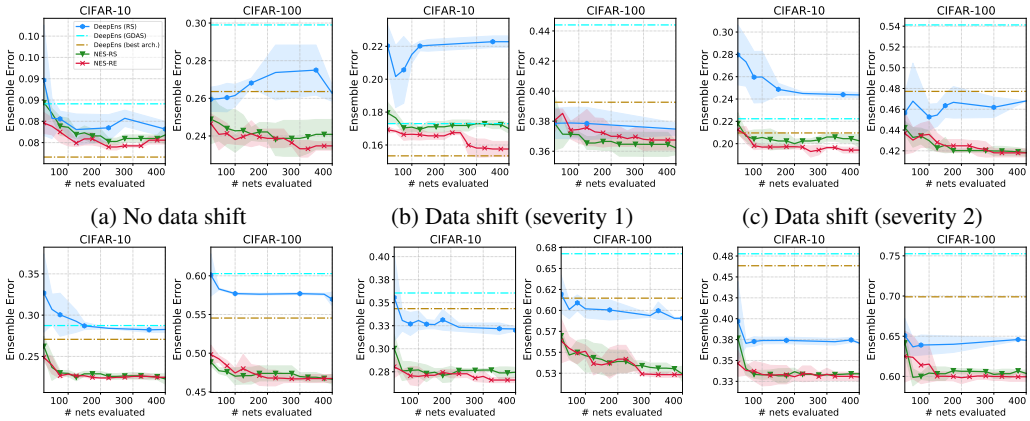


(a) No data shift (b) Data shift (severity 1)

(c) Data shift (severity 2) (d) Data shift (severity 3)

(e) Data shift (severity 4) (f) Data shift (severity 5)

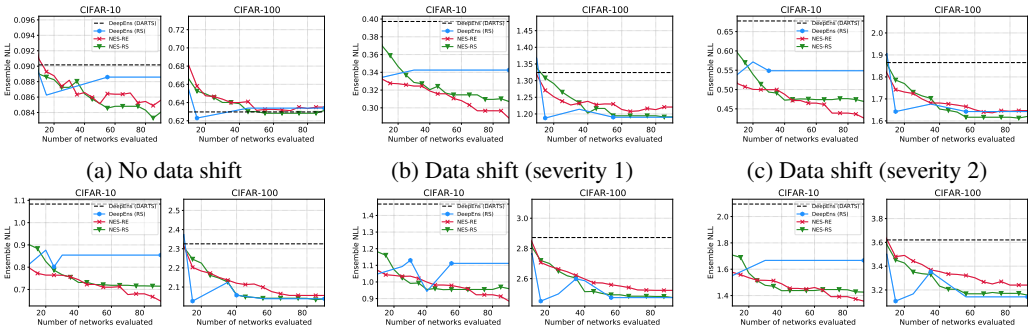
Figure 25: CIFAR-100 analogous to Figure 24.



(a) No data shift (b) Data shift (severity 1) (c) Data shift (severity 2)

(d) Data shift (severity 3) (e) Data shift (severity 4) (f) Data shift (severity 5)

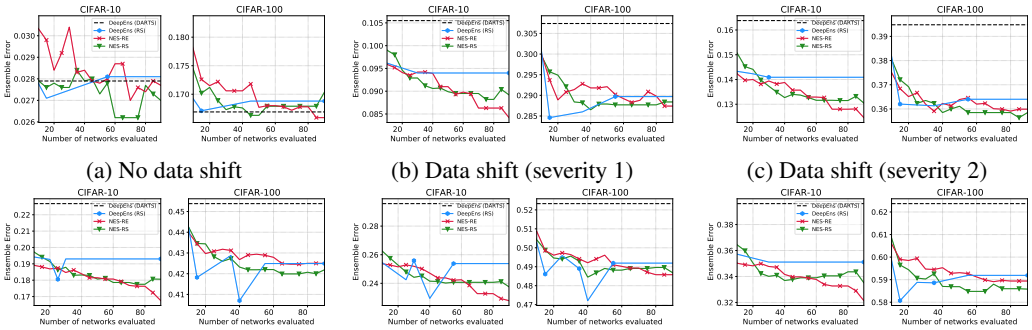
Figure 26: Ensemble error vs. budget K on the NAS-Bench-201 space (CIFAR-10 and CIFAR-100). Ensemble size fixed at $M = 3$.



(a) No data shift (b) Data shift (severity 1) (c) Data shift (severity 2)

(d) Data shift (severity 3) (e) Data shift (severity 4) (f) Data shift (severity 5)

Figure 27: High fidelity NLL vs. budget K on CIFAR-10 and CIFAR-100 with and without respective dataset shifts over the DARTS search space. Ensemble size is fixed at $M = 10$.



(a) No data shift (b) Data shift (severity 1) (c) Data shift (severity 2)

(d) Data shift (severity 3) (e) Data shift (severity 4) (f) Data shift (severity 5)

Figure 28: High fidelity classification error vs. budget K on CIFAR-10 and CIFAR-100 with and without respective dataset shifts over the DARTS search space. Ensemble size is fixed at $M = 10$.

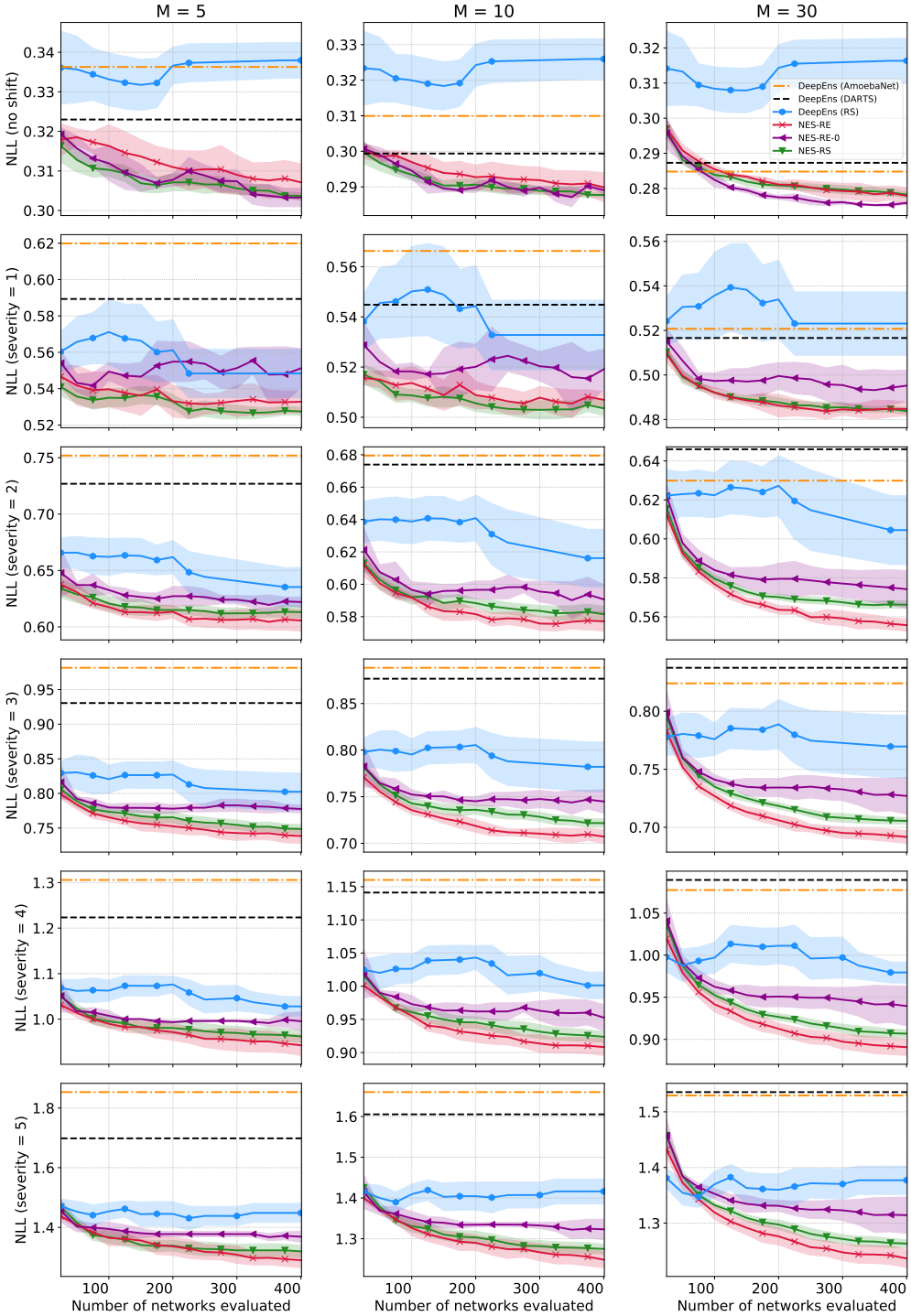
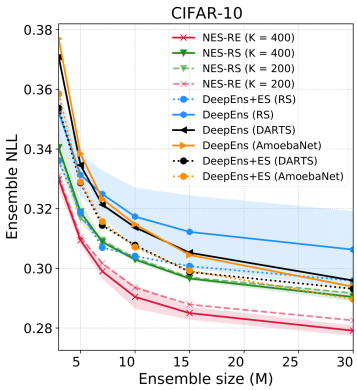


Figure 29: Results on CIFAR-10 [26] with varying ensembles sizes M and shift severity. Lines show the mean NLL achieved by the ensembles with 95% confidence intervals. See Appendix C.1.3 for the definition of NES-RE-0.



Tiny ImageNet

Method	No dataset shift		Dataset shift: severity 5	
	NLL	**Error (%)**	**NLL**	**Error (%)**
DeepEns (DARTS)	1.59	39.08	3.75	71.58
DeepEns + ES (DARTS)	1.57	38.68	3.68	70.90
DeepEns (AmoebaNet)	1.55	38.46	3.76	71.72
DeepEns + ES (AmoebaNet)	1.54	38.12	3.70	71.68
DeepEns (RS)	$1.51_{\pm 0.00}$	$37.46_{\pm 0.27}$	$3.60_{\pm 0.03}$	$70.48_{\pm 0.38}$
DeepEns + ES (RS)	1.50	36.98	3.55	**70.10**
NES-RS ($K = 200$)	$1.51_{\pm 0.01}$	$37.42_{\pm 0.21}$	$3.53_{\pm 0.01}$	$\mathbf{69.93}_{\pm 0.23}$
NES-RS ($K = 400$)	$1.48_{\pm 0.01}$	$37.45_{\pm 0.25}$	$\mathbf{3.51}_{\pm 0.01}$	$\mathbf{70.15}_{\pm 0.34}$
NES-RE ($K = 200$)	$1.48_{\pm 0.01}$	$36.98_{\pm 0.57}$	$3.55_{\pm 0.02}$	$70.22_{\pm 0.13}$
NES-RE ($K = 400$)	$\mathbf{1.44}_{\pm 0.02}$	$\mathbf{36.55}_{\pm 0.17}$	$3.54_{\pm 0.02}$	$\mathbf{69.98}_{\pm 0.24}$

Figure 30: Comparison of NES to deep ensembles with ensemble selection over initializations over the DARTS search space. LEFT: NLL vs ensemble size on CIFAR-10 (similar to ablation in Figure 7). RIGHT: Test NLL and classification error NES vs. deep ensembles on Tiny ImageNet with ensemble size $M = 10$.

C.1.4 Additional Results on the NAS-Bench-201 search space

In this section, we provide additional experimental results on CIFAR-10, CIFAR-100 and ImageNet-16-120 on the NAS-Bench-201 search space, complementing the results in Section 5 as shown in Figures 24-26.

C.2 Ablation study: NES-RE optimizing only on $\mathcal{D}_{\text{val}}$

We also include a variant of NES-RE, called NES-RE-0, in Figure 29. NES-RE and NES-RE-0 are the same, except that NES-RE-0 uses the validation set $\mathcal{D}_{\text{val}}$ without any shift during iterations of evolution, as in line 4 of Algorithm 1. Following the discussion in Appendix B.4, recall that this is unlike NES-RE, where we sample the validation set to be either $\mathcal{D}_{\text{val}}$ or $\mathcal{D}_{\text{val}}^{\text{shift}}$ at each iteration of evolution. Therefore, NES-RE-0 evolves the population without taking into account dataset shift, with $\mathcal{D}_{\text{val}}^{\text{shift}}$ only being used for the post-hoc ensemble selection step in line 9 of Algorithm 1.

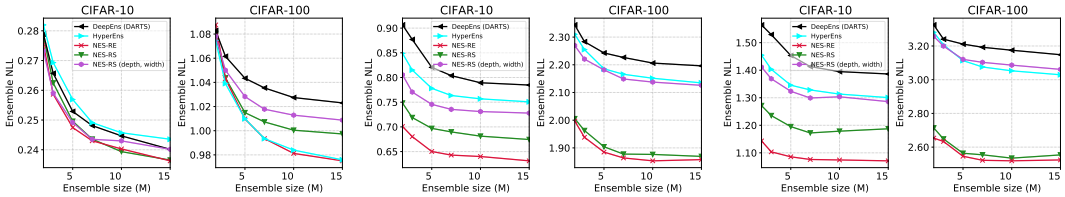
As shown in the Figure 29, NES-RE-0 shows a minor improvement over NES-RE in terms of loss for ensemble size $M = 30$ in the absence of dataset shift. This is in line with expectations, because evolution in NES-RE-0 focuses on finding base learners which form strong ensembles for in-distribution data. On the other hand, when there is dataset shift, the performance of NES-RE-0 ensembles degrades, yielding higher loss and error than both NES-RS and NES-RE. Nonetheless, NES-RE-0 still manages to outperform the DeepEns baselines consistently. We draw two conclusions on the basis of these results: (1) NES-RE-0 can be a competitive option in the absence of dataset shift. (2) Sampling the validation set, as done in NES-RE, to be $\mathcal{D}_{\text{val}}$ or $\mathcal{D}_{\text{val}}^{\text{shift}}$ in line 4 of Algorithm 1 plays an important role is returning a final pool $\mathcal{P}$ of base learners from which `ForwardSelect` can select ensembles robust to dataset shift.

C.3 What if deep ensembles use ensemble selection over initializations?

We provide additional experimental results in this section for comparing NES to deep ensembles with ensemble selection building on those shown in Figure 7 in Section 5. The table in Figure 30-right compares the methods with respect to classification error as well as loss at different dataset shift severities. In the absence of any dataset shift, we find that NES-RE outperforms all baselines with respect to both metrics, specially at the higher budget of $K = 400$ and NES-RS performs competitively. At shift severity 5, NES algorithms also outperform the baselines, with NES-RS performing slightly better than NES-RE. The plot in Figure 30-left shows similar results on CIFAR-10 over the DARTS search space for the same experiment as the one conducted on Tiny ImageNet and shown in Figure 7.

C.4 Comparing NES to ensembles with other varying hyperparameters

Since varying the architecture in an ensemble improves predictive performance and uncertainty estimation as demonstrated in Section 5, it is natural to ask what other hyperparameters should be



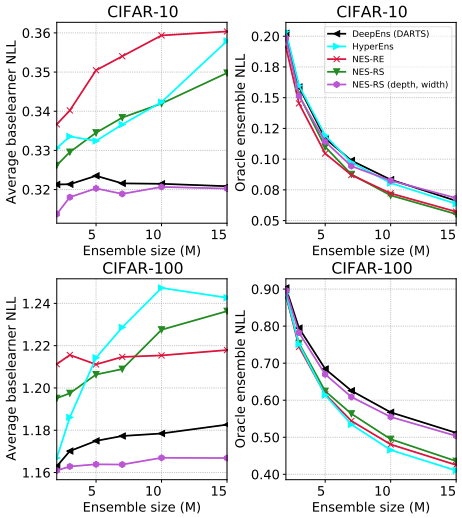
(a) No data shift (b) Data shift (severity 3) (c) Data shift (severity 5)

Figure 31: Plots show NLL vs. ensemble sizes comparing NES to the baselines introduced in Appendix C.4 on CIFAR-10 and CIFAR-100 with and without respective dataset shifts [26].

varied in an ensemble. It is also unclear which hyperparameters might be more important than others. Note that concurrent work by Wenzel *et al.* [53] has shown that varying hyperparameters such as L_2 regularization strength, dropout rate and label smoothing parameter also improves upon deep ensembles. While these questions lie outside the scope of our work and are left for future work, we conduct preliminary experiments to address them.

In this section, we consider two additional baselines working over the DARTS search space on CIFAR-10/100:

1. **HyperEns:** Optimize a fixed architecture, train K random initializations of it *where the learning rate and L_2 regularization strength are also sampled randomly* and select the final ensemble of size M from the pool using `ForwardSelect`. This is similar to `hyper ens` from Wenzel *et al.* [53].
2. **NES-RS (depth, width):** As described in Appendix B.1, NES navigates a complex (non-Euclidean) search space of architectures by varying the cell, which involves changing both the DAG structure of the cell and the operations at each edge of the DAG. We consider a baseline in which we keep the cell fixed (the optimized DARTS cell) and only vary the width and depth of the overall architecture. More specifically, we vary the number of *initial channels* $\in \{12, 14, 16, 18, 20\}$ (width) and the number of *layers* $\in \{5, 8, 11\}$ (depth). We apply NES-RS over this substantially simpler search space of architectures as usual: train K randomly sampled architectures (i.e. sampling only depth and width) to form a pool and select the ensemble from it.



(a) Average base learner loss and oracle ensemble loss for NES and the baselines introduced in Appendix C.4 on CIFAR-10 and CIFAR-100. Recall that small oracle ensemble loss generally corresponds to higher diversity.

Dataset	Shift Severity	DARTS search space				
		DeepEns (DARTS)	HyperEns	NES-RS (depth, width)	NES-RS	NES-RE
C10	0	8.2	8.1	8.0	8.0	**7.7**
	3	25.9	25.0	24.1	22.5	**21.5**
	5	43.3	40.8	42.1	38.1	**34.9**
C100	0	28.8	**28.1**	28.8	28.4	28.4
	3	54.0	52.7	53.1	48.9	**48.5**
	5	68.4	67.2	67.9	61.3	**60.7**

(b) Classification errors comparing NES to the baselines introduced in Appendix C.4 for different shift severities and $M = 10$. Best values are bold faced.

Figure 32: See Appendix C.4 for details.

The results shown in Figures 31 and Table 32b compare the two baselines above to DeepEns (DARTS), NES-RS and NES-RE.[6] As shown in Figure 31, NES-RE tends to outperform the baselines, though is at par with HyperEns on CIFAR-100 without dataset shift (Figure 31a). Under the presence of dataset shift (Figures 31b and 31c), both NES algorithms substantially outperform all baselines. Note that both HyperEns and NES-RS (depth, width) follow the same protocol as NES-RS and NES-RE: ensemble selection uses a shifted validation dataset when evaluating on a

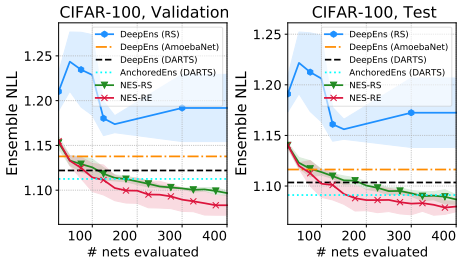
[6]Note that runs of DeepEns (DARTS), NES-RE and NES-RS differ slightly in this section relative to Section 5, as we tune the learning rate and L_2 regularization strength for each dataset instead of using the defaults used in Liu *et al.* [39]. This yields a fair comparison: HyperEns varies the learning rate and L_2 regularization while using a fixed, optimized architecture (DARTS), whereas NES varies the architecture while using fixed, optimized learning rate and L_2 regularization strength.

shifted test dataset. In terms of classification error, the observations are similar as shown in Table 32b. Lastly, we view the diversity of the ensembles from the perspective of oracle ensemble loss in Figure 32a. As in Section 5, results here also suggest that NES agorithms tend to find more diverse ensembles despite having higher average base learner loss.

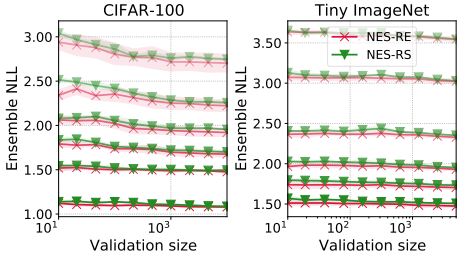
C.5 Sensitivity to the size of the validation dataset $\mathcal{D}_{\text{val}}$ and overfitting during ensemble selection

In this section, we consider how sensitive the performance of NES is to changes in the size of the validation dataset $\mathcal{D}_{\text{val}}$, and we discuss why overfitting to $\mathcal{D}_{\text{val}}$ is not a concern in our experiments. Recall that $\mathcal{D}_{\text{val}}$ is used by NES algorithms during ensemble selection from the pool of base learners as outlined in Algorithm 3, and we use $\mathcal{D}_{\text{val}}$ or $\mathcal{D}_{\text{val}}^{\text{shift}}$ depending on whether dataset shift is expected at test time as described in Section 4.3. Over the DARTS search space for CIFAR-100 and Tiny ImageNet, we measure the test loss of NES as a function of different validation dataset sizes. Specifically, we measure test loss of the ensembles selected using validation datasets of different sizes (with as few as 10 validation samples). Figure 33b shows this for different levels of dataset shift severity. The results indicate that NES is insensitive to the validation set size, achieving almost the same loss when using $10\times$ fewer validation data.

NES also did not overfit to the validation data in our experiments. This can be seen in Figure 33a for CIFAR-100 over the DARTS search space, which shows that both the test and validation losses decrease over time (i.e. as the pool size K increases). This finding is consistent across datasets and search spaces in our experiments. We also remark that overfitting is unexpected, because ensemble selection "fits a small number of parameters" since it only selects which base learners are added to the ensemble.



(a) Ensemble loss over the validation and test datasets over time (i.e. with increasing budget K). These curves being similar indicates no overfitting to the validation dataset during ensemble selection.



(b) Test performance of NES algorithms with varying validation data sizes. Each curve corresponds to one particular dataset shift severity (0-5 with 0 being no shift), with more transparent curves corresponding to higher shift severities. The loss stays relatively constant even with $10\times$ fewer validation data samples, indicating NES is not very sensitive to validation dataset size.

Figure 33: See Appendix C.5 for details.

C.6 Re-training the selected architectures on $\mathcal{D}_{\text{train}} + \mathcal{D}_{\text{val}}$

In practice, one might choose to re-trained the selected architectures from NES on the $\mathcal{D}_{\text{train}} + \mathcal{D}_{\text{val}}$ to make maximal use of the data available. We re-trained the ensembles constructed by NES and the best performing deep ensemble baseline, DeepEns (DARTS), on $\mathcal{D}_{\text{train}} + \mathcal{D}_{\text{val}}$, finding that both ensembles improve since the base learners improve due to more training data (see Figure 34). Note that, as shown in Appendix C.5, the performance of ensemble selection (which uses $\mathcal{D}_{\text{val}}$) is relatively insensitive to the size of $\mathcal{D}_{\text{val}}$. Therefore, one way to bypass the additional cost of having to re-train the ensembles on $\mathcal{D}_{\text{train}} + \mathcal{D}_{\text{val}}$ is to simply pick a very small $\mathcal{D}_{\text{val}}$, such that the performance of the models when trained on $\mathcal{D}_{\text{train}} + \mathcal{D}_{\text{val}}$ is approximately the same as when trained on $\mathcal{D}_{\text{train}}$.

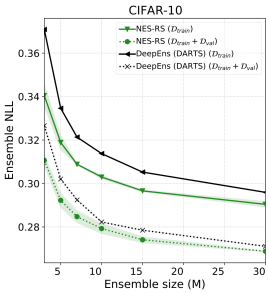


Figure 34: Re-training the selection architectures on $\mathcal{D}_{\text{train}} + \mathcal{D}_{\text{val}}$.

C.7 Averaging logits vs. averaging probabilities in an ensemble

In Figure 35, we explore whether ensembles should be constructed by averaging probabilities (i.e. post-softmax) as done in our work or by averaging the logits (i.e. pre-softmax). We find that while classification error of the resulting ensembles is similar, ensembles with averaged probabilities perform notably better in terms of uncertainty estimation (NLL) and are better calibrated as well (ECE).

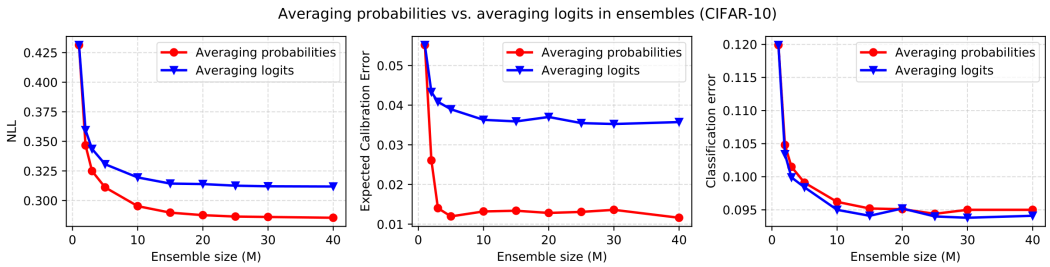
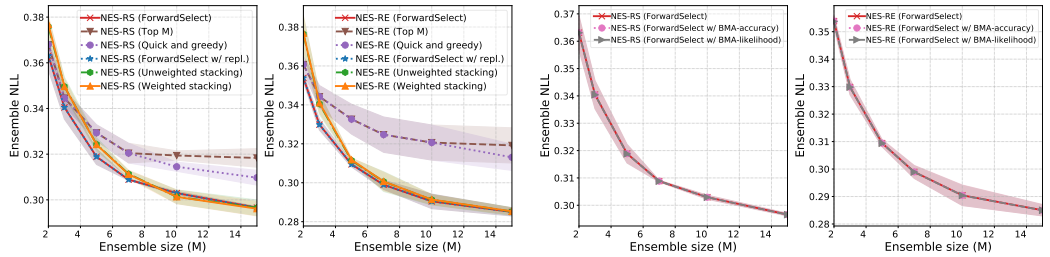


Figure 35: See Appendix C.7 for details.



(a) LEFT: NES-RS. RIGHT: NES-RE.

(b) LEFT: NES-RS. RIGHT: NES-RE.

Figure 36: `ForwardSelect` compared to other ensemble selection algorithms. Experiments were conducted on CIFAR-10 over the DARTS search space. See Appendix C.8 for details.

C.8 Comparison of ensemble selection algorithms

In Figure 9a, we compare the performance of NES for different choices of the ensemble selection algorithm (ESA) used in stage 2 (ensemble selection from pool). In addition to our default `ForwardSelect` without replacement, we experiment with the following seven choices:

- `TopM`: selects the top M (ensemble size) models by validation performance.
- `QuickAndGreedy`: Starting with the best network by validation performance, add the next best network to the ensemble only if it improves validation performance, iterating until the ensemble size is M or all models have been considered (returning an ensemble of size at most M).
- `ForwardSelect` with replacement: `ForwardSelect` but with replacement which allows repetitions of base learners.
- `Stacking` (Weighted): Linearly combines all base learners in the pool with learned stacking weights (which are positive and sum to 1). Then keeps only the base learners with the M largest stacking weights and weights them using the renormalized learned stacking weights.
- `Stacking` (Unweighted): Linearly combines all base learners in the pool with learned stacking weights (which are positive and sum to 1). Then keeps only the base learners with the M largest stacking weights and combines them by a simple, unweighted average.
- `ForwardSelect` with Bayesian model averaging by likelihood: Selects the base learners in the ensemble using `ForwardSelect` and then takes an average weighted by the (normalized) validation likelihoods of the base learners.
- `ForwardSelect` with Bayesian model averaging by accuracy: Selects the base learners in the ensemble using `ForwardSelect` and then takes an average weighted by the (normalized) validation accuracies of the base learners.

In Figure 36 we provide additional results including the one in Figure 9a. From the plots we can observe that:

- `ForwardSelect` performs better than or at par with all ESAs considered here.
- `Stacking` tends to perform competitively but still worse than `ForwardSelect`, and whether weighted averaging is used or not has a very minor impact on performance, since the learned weights tend to be close to uniform.

- Bayesian model averaging (BMA) also has a very minor impact (almost invisible in the plots in Figure 36b), because the likelihoods and accuracies of individual base learners are very similar (a consequence of multi-modal loss landscapes in neural networks with different models achieving similar losses) hence the BMA weights are close to uniform. The NLL achieved by the BMA ensemble only differed at the 4th or 5th decimal places compared to unweighted ensembles with the same base learners.

Lastly, we assess the impact of *explicitly* regularizing for diversity during ensemble selection as follows: we use `ForwardSelect` as before but instead use it to minimize the objective "validation loss - diversity strength $\times$ diversity" where diversity is defined as the average (across base learners and data points) L_2 distance between a base learner's predicted class probabilities and the ensemble's predicted class probabilities. The plots below show the results for different choices of the "diversity strength" hyperparameter. In summary, if appropriately tuned, `ForwardSelect` with diversity performs slightly better than usual `ForwardSelect` for NES-RS, though for NES-RE, the diversity term seems to harm performance as shown in Figure 37-right.

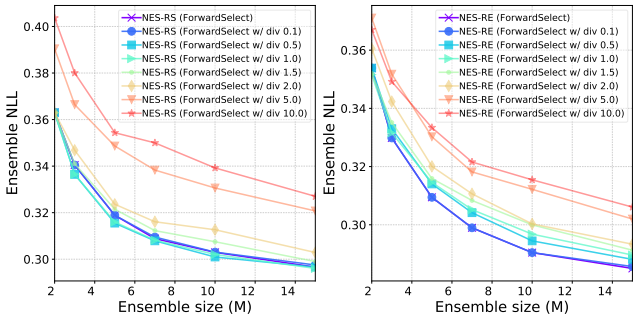


Figure 37: Ensemble selection with explicit diversity regularization. Experiments were conducted on CIFAR-10 over the DARTS search space.

C.9 Generating the architecture pool using weight-sharing NAS algorithms

In order to accelerate the NES search phase, we generated the pool using the weight sharing schemes proposed by Random Search with Weight Sharing [37] and DARTS [39]. Specifically, we trained one-shot weight-sharing models using each of these two algorithms, then we sampled architectures from the weight-shared models uniformly at random to build the pool. Next, we ran `ForwardSelect` as usual to select ensembles from these pools. Finally, we retrained the selected base learners from scratch using the same training pipeline as NES and the deep ensemble baselines. In Figure 38, we refer to the results of these methods as NES-RS-WS and NES-RS-DARTS, respectively. As we can see, their performance is worse than NES-RS and NES-RE, even though the computation cost is reduced significantly. This is likely due to the low correlation between the performance of the architectures when evaluated using the shared weights and the performance when re-trained in isolation. Prior work has shown this is caused by weight interference and co-adaptation which occurs during the weight-sharing model training [58, 59]. Nonetheless, we believe this is a promising avenue for future research requiring more development, as it substantially reduces the cost of exploring the search space.

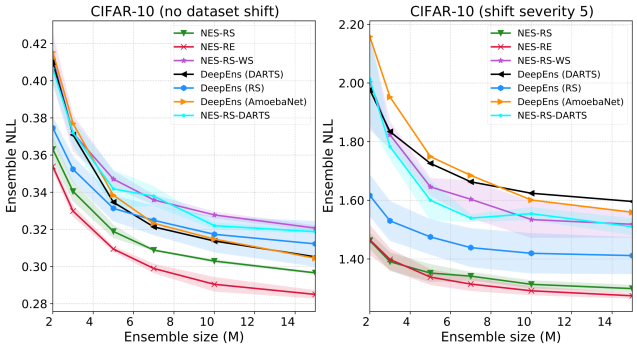


Figure 38: Comparison of NES-RS-WS and NES-RS-DARTS with the other baselines and NES-RS and NES-RE. The plots show the NLL vs. ensemble sizes on CIFAR-10 with and without dataset shifts over the DARTS search space. Mean NLL shown with 95 confidence intervals.

D NAS Best Practice Checklist

Our experimental setup resembles that of NAS papers, therefore, to foster reproducibility, we now describe how we addressed the individual points of the NAS best practice checklist by Lindauer & Hutter [38].

1. **Best Practices for Releasing Code**

   For all experiments you report:

   (a) Did you release code for the training pipeline used to evaluate the final architectures? [Yes] Code for our experiments, including NES and baselines, is open-source.

   (b) Did you release code for the search space? [Yes]

   (c) Did you release the hyperparameters used for the final evaluation pipeline, as well as random seeds? [Yes]

   (d) Did you release code for your NAS method? [Yes]

   (e) Did you release hyperparameters for your NAS method, as well as random seeds? [Yes]

2. **Best practices for comparing NAS methods**

   (a) For all NAS methods you compare, did you use exactly the same NAS benchmark, including the same dataset (with the same training-test split), search space and code for training the architectures and hyperparameters for that code? [Yes] All base learners follow the same training routine and use architectures from the same search space.

   (b) Did you control for confounding factors (different hardware, versions of DL libraries, different runtimes for the different methods)? [Yes]

   (c) Did you run ablation studies? [Yes] See Section 5.2 and Appendices C.2, C.3, C.4, C.5, C.6, C.7, C.8, C.9.

   (d) Did you use the same evaluation protocol for the methods being compared? [Yes]

   (e) Did you compare performance over time? [Yes] As shown in all figures with K (which is proportional to training time) on the x-axis.

   (f) Did you compare to random search? [Yes]

   (g) Did you perform multiple runs of your experiments and report seeds? [Yes] We ran NES with multiple seeds and averaged over them in Section 5.

   (h) Did you use tabular or surrogate benchmarks for in-depth evaluations? [Yes] We performed experiments over the tabular NAS-Bench-201 search space, including an in-depth evaluation of NES by comparing it to the deep ensemble of the best architecture in the search space.

3. **Best practices for reporting important details**

   (a) Did you report how you tuned hyperparameters, and what time and resources this required? [Yes] We relied on default hyperparameter choices from Liu *et al.* [39] as described in Appendix B.

   (b) Did you report the time for the entire end-to-end NAS method (rather than, e.g., only for the search phase)? [Yes] See Figure 7 and Appendix B.3.

   (c) Did you report all the details of your experimental setup? [Yes]