## 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 $F_{\text{OE}}$, 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, $f_{\theta_k}$ assigns the highest probability to the correct class $y$ for input $\boldsymbol{x}$. Since $-\log$ is a decreasing function, we have

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

We apply Jensen's inequality in its finite form for the second inequality. Jensen's inequality states that for a real-valued, convex function $\varphi$ 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. Note that unless stated otherwise, all sampling over a discrete set is done uniformly in the discussion below.

### B.1 Architecture Search Spaces

**DARTS search space.** The first architecture search space we consider in our experiments is the one from DARTS (Liu et al., 2019). We search for two types of *cells*: *normal* cells, which preserve the spatial dimensions, and *reduction* cells, which reduce the spatial dimensions. These cells are stacked using a pre-determined macro-architecture where they are usually repeated and connected using additional skip connections. 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 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. 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. (2019) for more details.

**NAS-Bench-201 search space.** NAS-Bench-201 (Dong & Yang, 2020) 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 (2020).

### B.2 Datasets

**Fashion-MNIST (Xiao et al., 2017).** 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 (Krizhevsky et al., 2009).** CIFAR-10 and CIFAR-100 both consist of 60k $32\times32$ 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 (Le & Yang, 2015).** Tiny Imagenet has 200 classes and each class has 500 training, 50 validation and 50 test colour images with $64\times64$ 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 (Dong & Yang, 2020)** This variant of the ImageNet-16-120 (Chrabaszcz et al., 2017) contains 151.7k train, 3k validation and 3k test ImageNet images downsampled to $16\times16$ 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 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, 2019)) 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 (2019), 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, 2019). 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

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 and CIFAR-100 and 15 epochs for Fashion-MNIST. For Tiny ImageNet, we used a batch size of 128 for 100 epochs. Unlike DARTS, we do not use any data augmentation procedure during training, nor any additional regularization such as ScheduledDropPath (Zoph et al., 2018) or auxiliary heads, except for the case of Tiny ImageNet, for which we used ScheduledDropPath and standard data augmentation as default in DARTS. All other hyperparameter settings are exactly as in DARTS (Liu et al., 2019).

All results shown are averaged over multiple runs with error bars indicating a 95% confidence interval. We used a budget $K = 400$ in all experiments, except Tiny ImageNet on the DARTS search space, which used $K = 200$ and ImageNet-16-120 on the NAS-Bench-201 search space, which used $K = 1000$.

### 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 2 over multiple GPUs, called worker nodes. We use the parallelism scheme provided by the `hpbandster` (Falkner et al., 2018) codebase.[3] In brief, the master node keeps track of the population and history (lines 1, 4-6, 8 in Algorithm 2), and it distributes the training of the networks to the individual worker nodes (lines 2, 7 in Algorithm 2). 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:

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

- `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.
- `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.

See Real et al. (2019) for details and illustrations of these mutations. Note that for NAS-Bench-201, following Dong & Yang (2020) 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 2 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 $s \in \{0, 1, \dots, 5\}$, we apply `ForwardSelect` with $\mathcal{D}_{\text{val}}^{\text{shift}}$ of severity $s$ to select an ensemble from $\mathcal{P}$ (line 9 in Algorithm 2), 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 .

## 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 Additional Results on the DARTS search space

#### C.1.1 Results on Fashion-MNIST

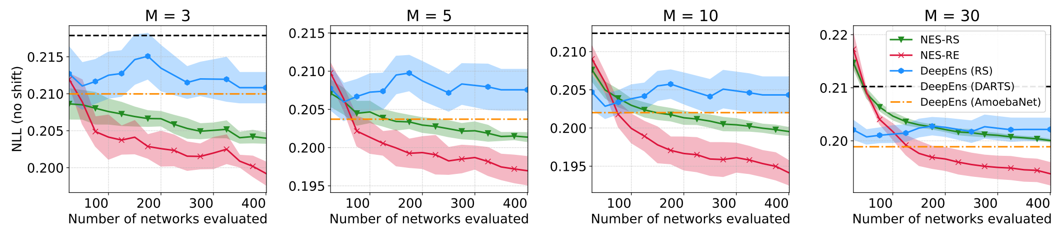


Figure 7: Results on Fashion-MNIST with varying ensembles sizes $M$. Lines show the mean NLL achieved by the ensembles with 95% confidence intervals.

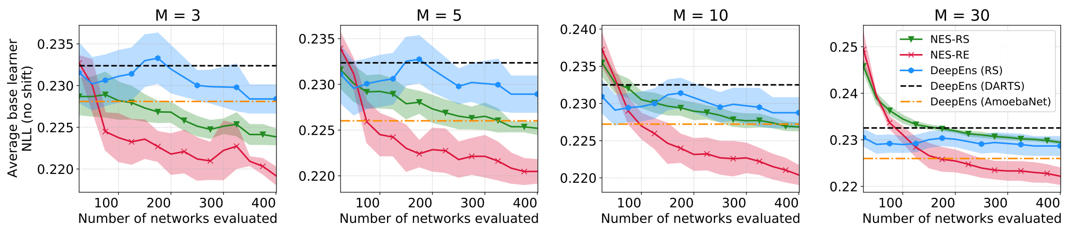


Figure 8: Average base learner loss for NES-RS, NES-RE and DeepEns (RS) on Fashion-MNIST. Lines show the mean NLL and 95% confidence intervals.

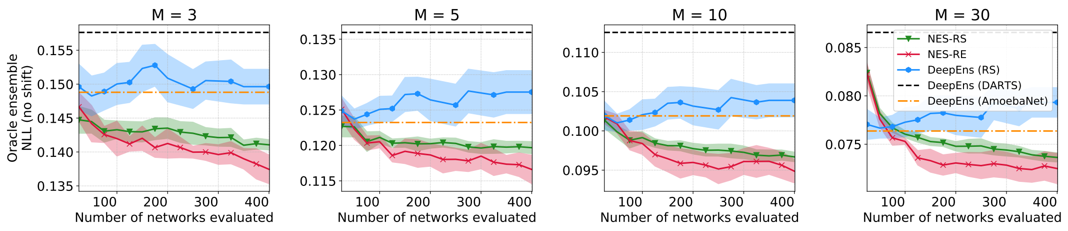


Figure 9: Oracle ensemble loss for NES-RS, NES-RE and DeepEns (RS) on Fashion-MNIST. Lines show the mean NLL and 95% confidence intervals.

As shown in Figure 7, 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 (Xiao et al., 2017), we compare the average base learner loss (Figure 8) and oracle ensemble loss (Figure 9) 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 8 and 9, 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 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 2, 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.

Table 2: 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.

| | **Classification Error** (out of 1) | | | | | **Expected Calibration Error (ECE)** | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| $M$ | 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** |

### 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 10, 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.

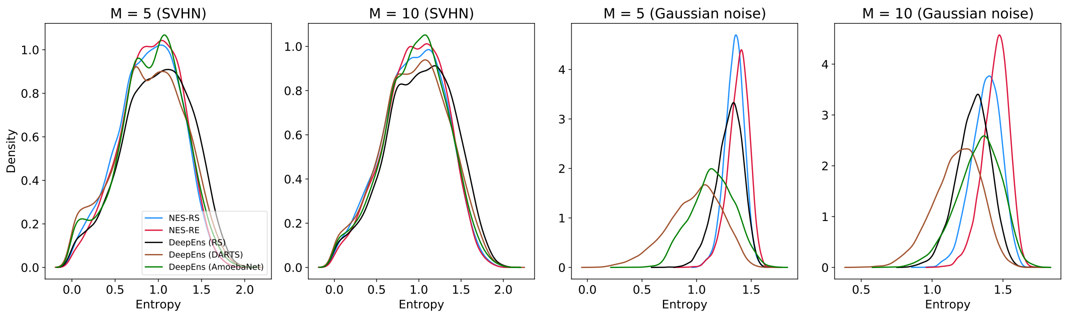


Figure 10: Entropy of predicted probabilities when trained on CIFAR-10 over the DARTS search space.

### C.1.3 Additional Results on CIFAR-10, CIFAR-100 and Tiny ImageNet

In this section, we provide additional experimental results on CIFAR-10, CIFAR-100 and Tiny ImageNet on the DARTS search space, complimenting the results in Section 5 as shown in Figures 12-18.

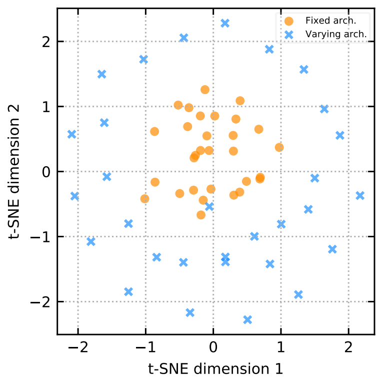


Figure 11: t-SNE visualization: predictions of base learners in two ensembles, one with fixed architecture and one with varying architectures.

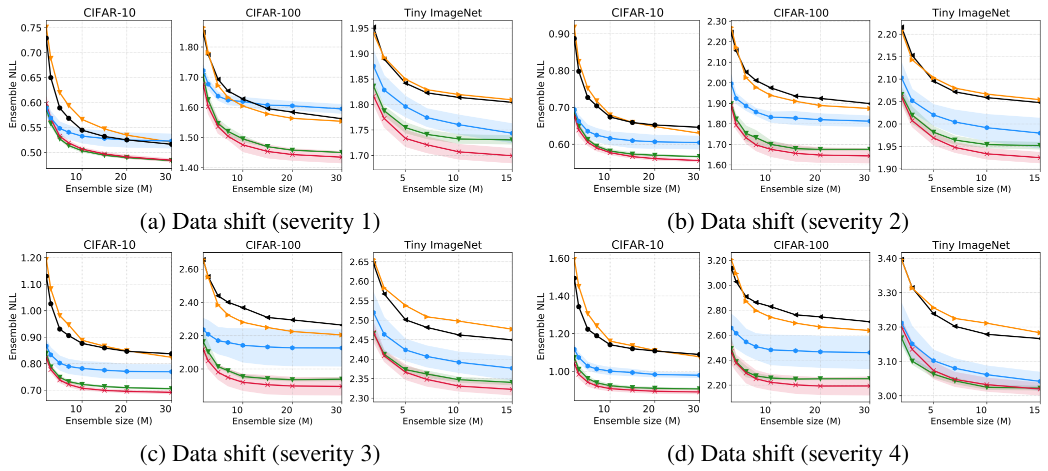
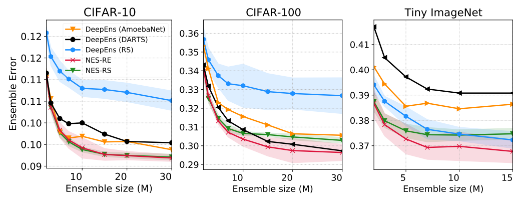
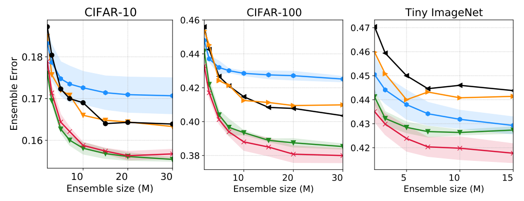


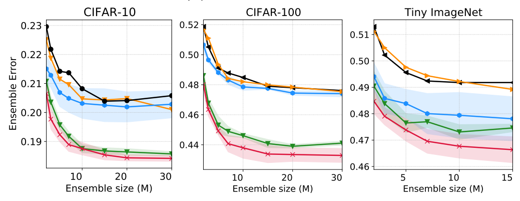
Figure 12: NLL vs. ensemble sizes on CIFAR-10, CIFAR-100 and Tiny ImageNet with varying dataset shifts (Hendrycks & Dietterich, 2019) over DARTS search space.



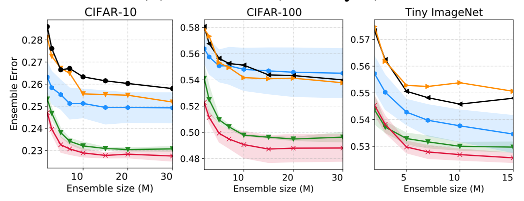
(a) No data shift



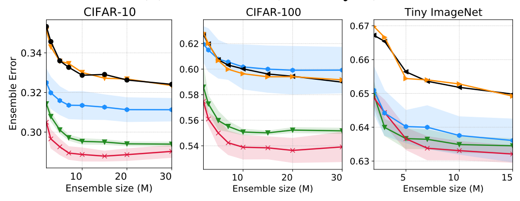
(b) Data shift (severity 1)



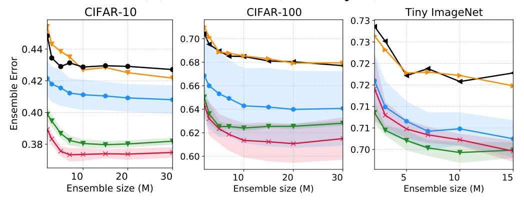
(c) Data shift (severity 2)



(d) Data shift (severity 3)

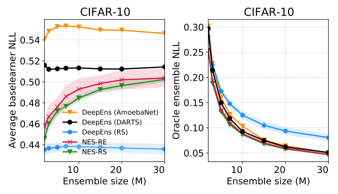


(e) Data shift (severity 4)

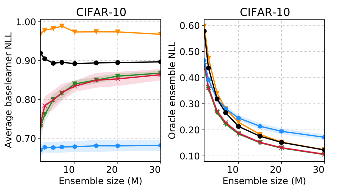


(f) Data shift (severity 5)

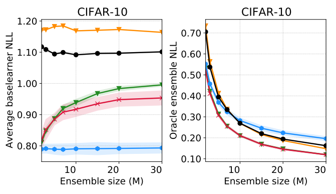
Figure 13: Classification error rate (between 0-1) vs. ensemble size on DARTS search space.



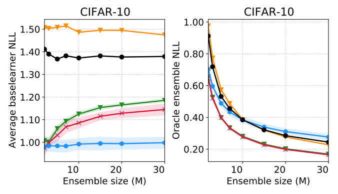
(a) No data shift



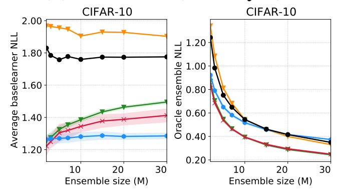
(b) Data shift (severity 1)



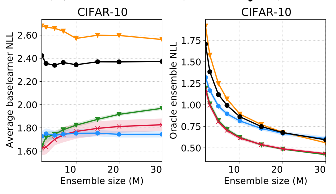
(c) Data shift (severity 2)



(d) Data shift (severity 3)

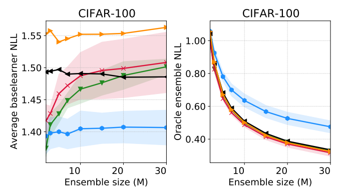


(e) Data shift (severity 4)

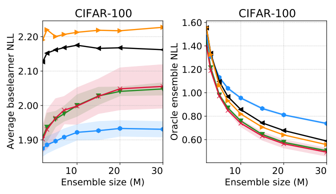


(f) Data shift (severity 5)

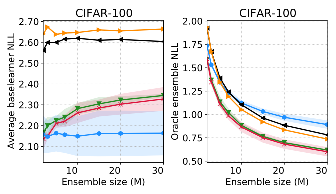
Figure 14: Average base learner and oracle ensemble NLL across ensemble sizes and shift severities on CIFAR-10 over DARTS search space.



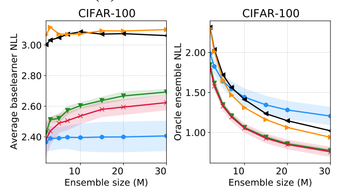
(a) No data shift



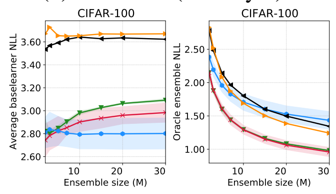
(b) Data shift (severity 1)



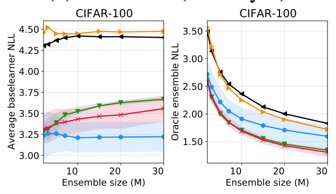
(c) Data shift (severity 2)



(d) Data shift (severity 3)

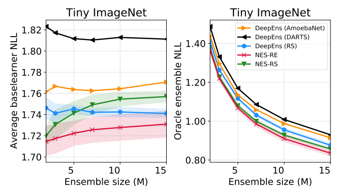


(e) Data shift (severity 4)

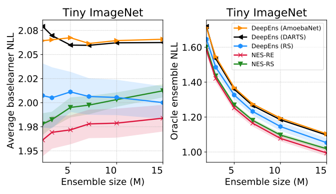


(f) Data shift (severity 5)

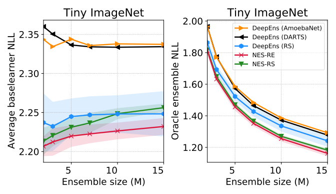
Figure 15: Average base learner and oracle ensemble NLL across ensemble sizes and shift severities on CIFAR-100 over DARTS search space.



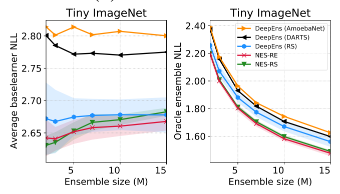
(a) No data shift



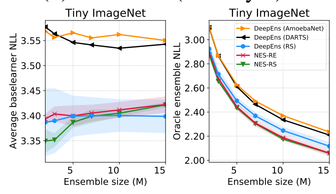
(b) Data shift (severity 1)



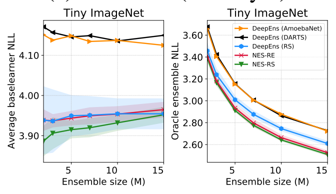
(c) Data shift (severity 2)



(d) Data shift (severity 3)

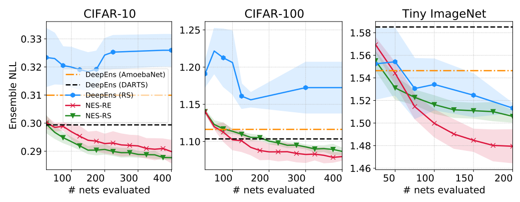


(e) Data shift (severity 4)

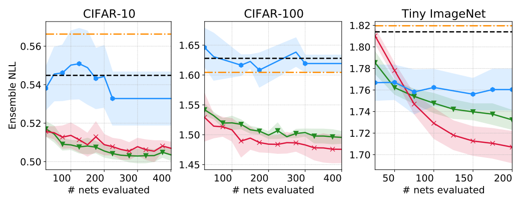


(f) Data shift (severity 5)

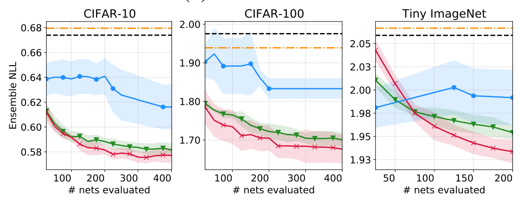
Figure 16: Average base learner and oracle ensemble NLL across ensemble sizes and shift severities on Tiny ImageNet over DARTS search space.



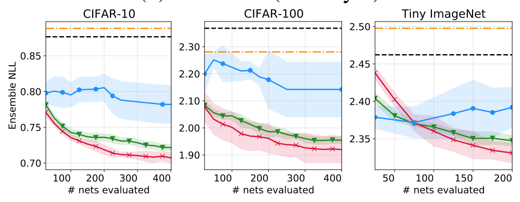
(a) No data shift



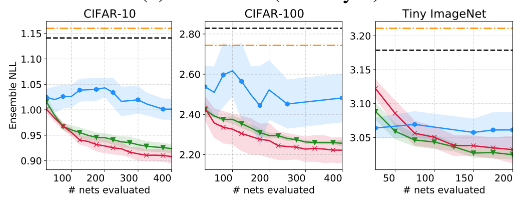
(b) Data shift (severity 1)



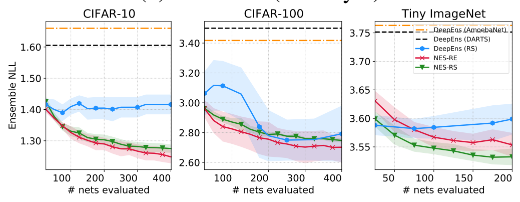
(c) Data shift (severity 2)



(d) Data shift (severity 3)

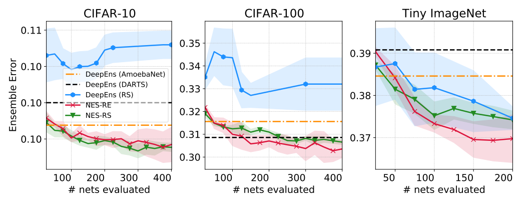


(e) Data shift (severity 4)

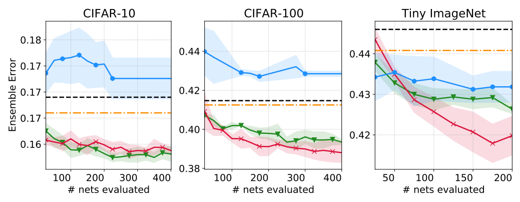


(f) Data shift (severity 5)

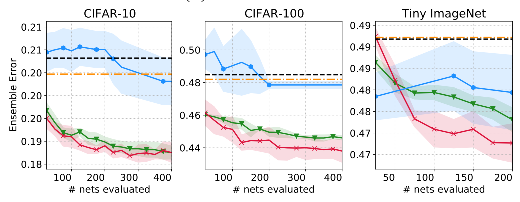
Figure 17: Ensemble NLL vs. budget $K$. Ensemble size fixed at $M = 10$.



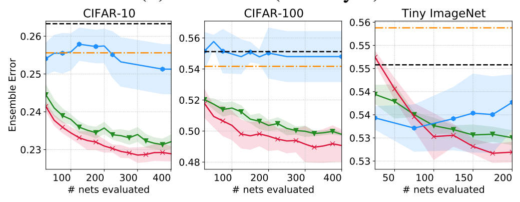
(a) No data shift



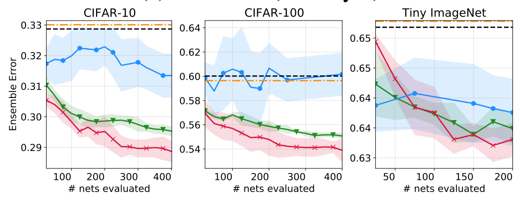
(b) Data shift (severity 1)



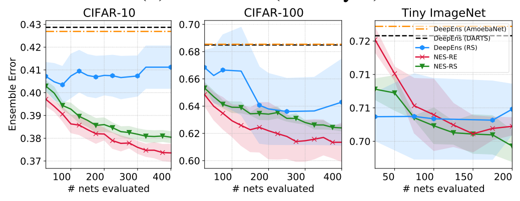
(c) Data shift (severity 2)



(d) Data shift (severity 3)



(e) Data shift (severity 4)



(f) Data shift (severity 5)

Figure 18: Ensemble error vs. budget $K$. Ensemble size fixed at $M = 10$.

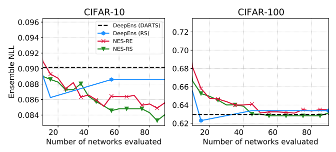
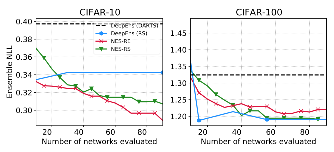
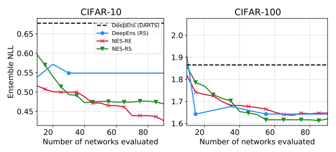
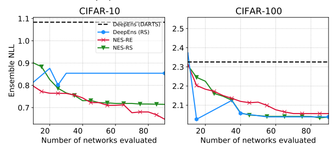
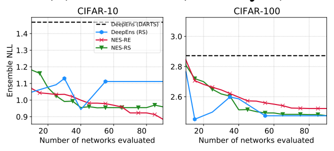
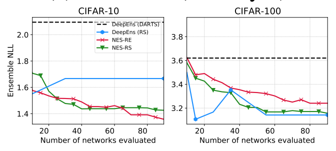

(a) No data shift

(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 19: 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 $M = 10$.

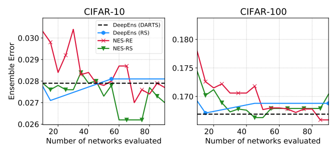
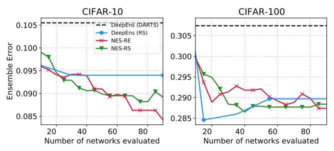
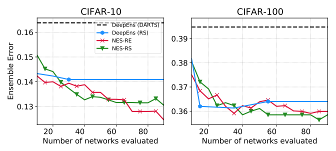
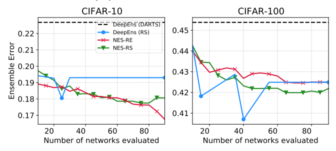
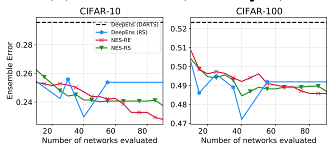
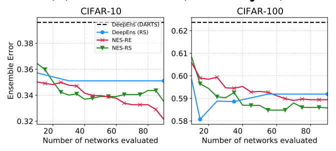

(a) No data shift

(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 20: 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 $M = 10$.

**Results on CIFAR for larger models.** In additional to the results on CIFAR-10 and CIFAR-100 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 19 and 20 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.

### C.2 Ablation study: NES-RE optimizing only on clean data

We also include a variant of NES-RE, called NES-RE-0, in Figure 21.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 2. 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 2.

As shown in the Figure 21, 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 2 plays an important role is returning a final pool $\mathcal{P}$ of base learners from which `ForwardSelect` can select ensembles robust to dataset shift.

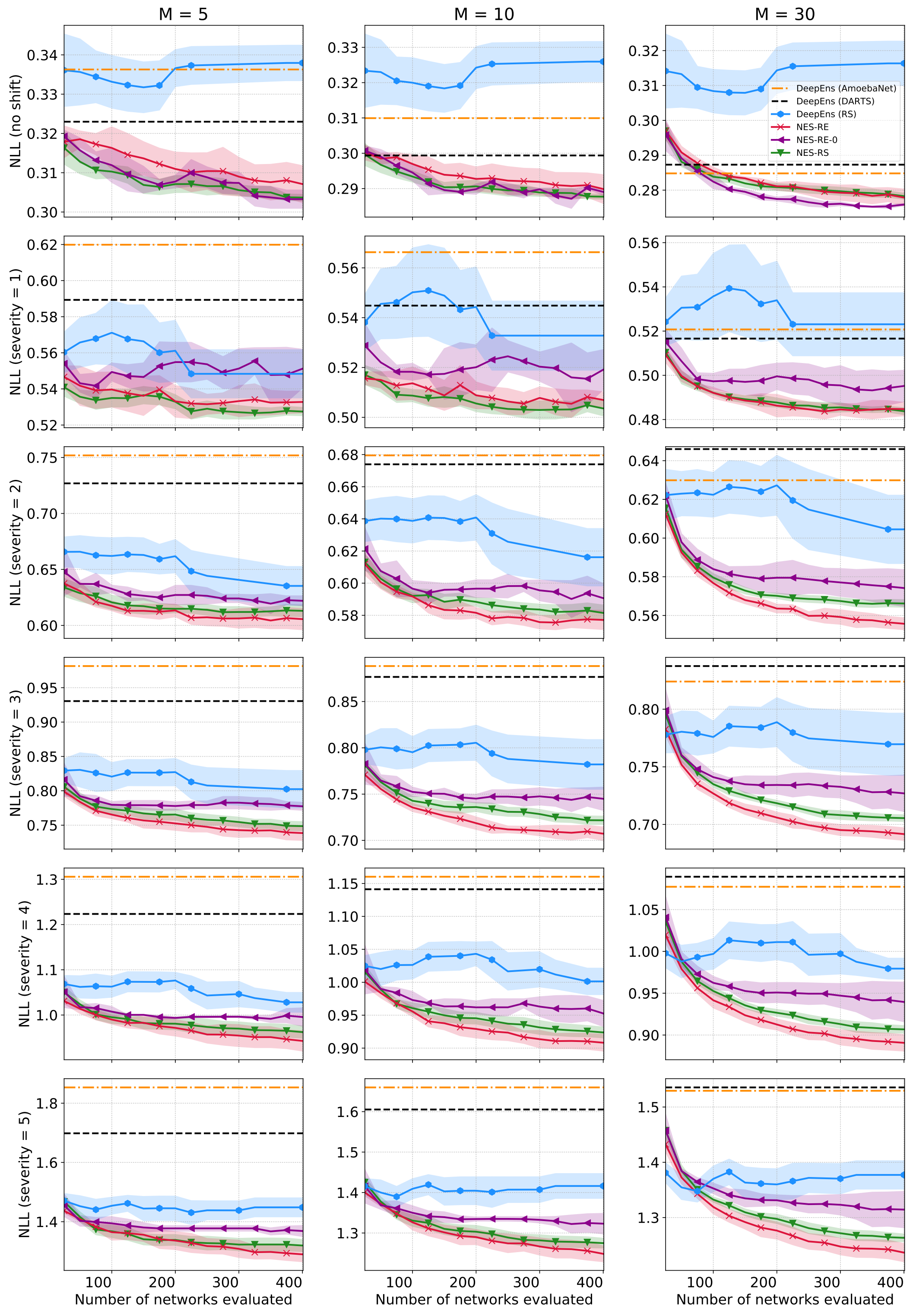


Figure 21: Results on CIFAR-10 (Hendrycks & Dietterich, 2019) with varying ensembles sizes $M$ and shift severity. Lines show the mean NLL achieved by the ensembles with 95% confidence intervals. See Appendix C.1.3 for the definition of NES-RE-0.

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

Recall that NES algorithms differ from deep ensembles in two important ways: the ensembles use varying architectures and NES utilizes ensemble selection (i.e. `ForwardSelect` applied to $\mathcal{P}$) to pick the base learners. In this section, we conduct a study intended to investigate the following question: is the improvement offered by NES over deep ensembles only due to ensemble selection? In other words, we wish to isolate and understand the impact of varying architectures by comparing NES to a baseline that also incorporates ensemble selection into the construction of deep ensembles.

Using the DARTS search space on Tiny ImageNet, we empirically compare NES to the baselines "DeepEns + ES" which 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 $M$ from the pool. This yields the three additional baselines DeepEns + ES (DARTS/AmoebaNet/RS) which correspond to optimizing the fixed architectures using the DARTS algorithm (DARTS), regularized evolution (AmoebaNet) and random search (RS).

The results indicate that NES outperforms or is at par with DeepEns + ES baselines, as shown in Table 3 and Figure 22. In particular, both NES algorithms outperform DeepEns + ES (DARTS/AmoebaNet). DeepEns + ES (RS) is the most competitive of the deep ensemble baselines, which is improved upon by NES-RE and is competitive with NES-RS. Also, as expected, deep ensembles *with* ensemble selection consistently perform better than their counterparts *without* ensemble selection.

Table 3 also includes the computational costs of each method measured in terms of the number of networks trained. For each DeepEns + ES baseline, we used a pool of size $K = 200$ (as with NES) from which the ensemble is selected. This cost comes in addition to the cost of optimizing the fixed base learner architecture prior to forming the pool. For instance, the architecture for DeepEns + ES (RS) is optimized by random search, selecting the best architecture by validation loss from a random sample of $K = 200$ (trained) architectures; this yields a total cost of 400 networks trained. This is twice the cost of NES algorithms which required training 200 architectures to form the pool.

### 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 varied in an ensemble. It is also unclear which hyperparameters might be more important than others. Note that concurrent work by Wenzel et al. (2020) 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.

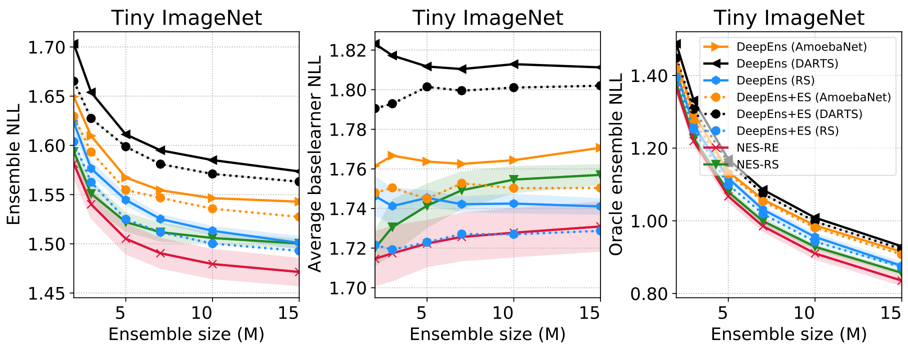


Figure 22: Loss vs. ensemble size for NES and deep ensembles (with/without ensemble selection over initializations). The left plot shows that NES-RE outperforms all other methods across ensemble sizes. The right plot shows that ensembles produced by NES algorithms also consistently have higher diversity (as indicated by smaller oracle ensemble loss). See Appendix C.3 for details.

Table 3: A comparison of NES to deep ensembles with ensemble selection over initializations for Tiny ImageNet over the DARTS search space with ensemble size $M = 10$. The computational costs are reports in terms of the number of networks trained (a typical network from this search space takes 3 hours to train on an NVIDIA RTX 2080Ti). The "arch" column indicates the number of architectures evaluated to find the architecture and the "ensemble" column indicates the number of architectures evaluated for building the ensemble. Note that for DARTS and AmoebaNet we convert the GPU hours for finding the architecture into number of networks trained by dividing by 3. See Appendix C.3 for details.

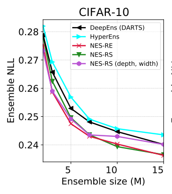
| **Method** | No dataset shift | | Dataset shift (severity 5) | | **Cost (# nets trained)** | |
|---|---|---|---|---|---|---|
| | **NLL** | **Classif. Error (%)** | **NLL** | **Classif. Error (%)** | **Arch.** | **Ensemble** |
| DeepEns (DARTS) | 1.59 | 39.08 | 3.75 | 71.58 | 32 | 10 |
| DeepEns + ES (DARTS) | 1.57 | 38.68 | 3.68 | 70.90 | 32 | 200 |
| DeepEns (AmoebaNet) | 1.55 | 38.46 | 3.76 | 71.72 | 25200 | 10 |
| DeepEns + ES (AmoebaNet) | 1.54 | 38.12 | 3.70 | 71.68 | 25200 | 200 |
| DeepEns (RS) | $1.51_{\pm 0.00}$ | $\mathbf{37.46}_{\pm 0.27}$ | $3.60_{\pm 0.03}$ | $70.48_{\pm 0.38}$ | 200 | 10 |
| DeepEns + ES (RS) | 1.50 | **36.98** | 3.55 | **70.10** | 200 | 200 |
| NES-RS | $1.51_{\pm 0.01}$ | $\mathbf{37.42}_{\pm 0.21}$ | $\mathbf{3.53}_{\pm 0.01}$ | $\mathbf{69.93}_{\pm 0.23}$ | 200 | |
| NES-RE | $\mathbf{1.48}_{\pm 0.01}$ | $\mathbf{36.98}_{\pm 0.57}$ | $3.55_{\pm 0.02}$ | $70.22_{\pm 0.13}$ | 200 | |

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

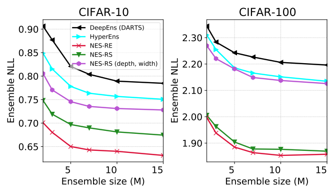
1. **HyperEns:** Optimize a fixed architecture, train $K$ random initializations of it *where the learning rate and $L_2$ regularization strength are also sampled randomly* and select the final ensemble of size $M$ from the pool using `ForwardSelect`. This is similar to `hyper ens` from Wenzel et al. (2020).
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.

The results shown in Figures 23 and Table 4 compare the two baselines above to DeepEns (DARTS), NES-RS and NES-RE.[4] As shown in Figure 23, NES-RE tends to outperform the baselines, though is at par with HyperEns on CIFAR-100 without dataset shift (Figure 23a). Under the presence of dataset shift (Figures 23b and 23c), 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 shifted test dataset. In terms of classification error, the observations are similar as shown in Table 4. Lastly, we view the diversity of the ensembles from the perspective of oracle ensemble loss in Figure 24. As in Section 5, results here also suggest that NES agorithms tend to find more diverse ensembles despite having higher average base learner loss.

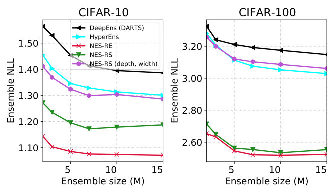
[4]Note that runs of DeepEns (DARTS), NES-RE and NES-RS differ slightly in this section relative to Section 5, as we tune the learning rate and $L_2$ regularization strength for each dataset instead of using the defaults used in Liu et al. (2019). This yields a fair comparison: HyperEns varies the learning 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.



(a) No data shift



(b) Data shift (severity 3)



(c) Data shift (severity 5)

Figure 23: 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 (Hendrycks & Dietterich, 2019).

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

| **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** |

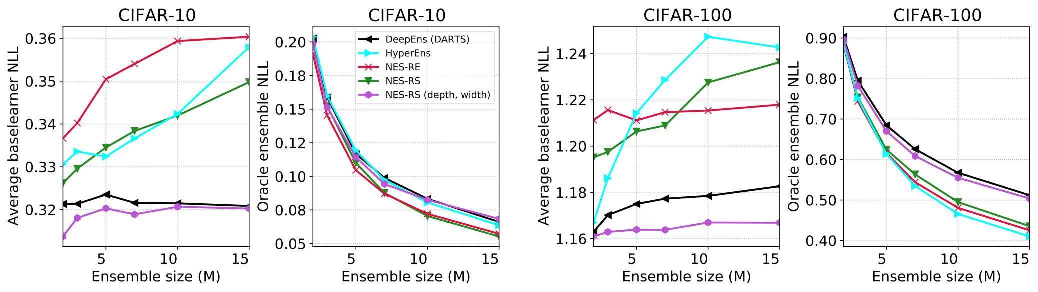


Figure 24: Average base learner loss and oracle ensemble loss for NES and the baselines introduced in Appendix C.4 on CIFAR-10 and CIFAR-100. Recall that small oracle ensemble loss generally corresponds to higher diversity.