

---

# Multi-headed Neural Ensemble Search

---

Ashwin Raaghav Narayanan<sup>1</sup> Arbër Zela<sup>1</sup> Tonmoy Saikia<sup>1</sup> Thomas Brox<sup>1</sup> Frank Hutter<sup>1,2</sup>

## Abstract

Ensembles of CNN models trained with different seeds (also known as Deep Ensembles) are known to achieve superior performance over a single copy of the CNN. Neural Ensemble Search (NES) can further boost performance by adding architectural diversity. However, the scope of NES remains prohibitive under limited computational resources. In this work, we extend NES to multi-headed ensembles, which consist of a shared backbone attached to multiple prediction heads. Unlike Deep Ensembles, these multi-headed ensembles can be trained end to end, which enables us to leverage one-shot NAS methods to optimize an ensemble objective. With extensive empirical evaluations, we demonstrate that multi-headed ensemble search finds robust ensembles 3× faster, while having comparable performance to other ensemble search methods, in both predictive performance and uncertainty calibration.

## 1. Introduction

Ensembles of neural networks are a common solution to improve predictive performance and uncertainty calibration (Hansen & Salamon, 1990). Ensembles of networks trained with different random initializations (known as *Deep Ensembles*) can lead to well-performing and diverse models as they include different local optima (Lakshminarayanan et al., 2016; Fort et al., 2019). They have also been shown to outperform approximate Bayesian methods (Lakshminarayanan et al., 2016; Ovadia et al., 2019; Gustafsson et al., 2020). On top of that, Neural Ensemble Search (NES) (Zaidi et al., 2020) and HyperDeepEns (Wenzel et al., 2021) can further improve performance by searching for a complementary set of ensemble members. However, building and maintaining multiple models can be expensive.

<sup>1</sup>University of Freiburg <sup>2</sup>Bosch Center for Artificial Intelligence. Correspondence to: Ashwin Raaghav Narayanan <anarayan@cs.uni-freiburg.de>, Arbër Zela <zela@cs.uni-freiburg.de>, Tonmoy Saikia <saikiat@cs.uni-freiburg.de>.

Table 1. Overview: Multi-headed NES (MH-NES) leads to performant models on CIFAR-100 with significantly less search costs.

Method	# Params (M)	Search Cost (GPU hours)	Error	ECE
NES-RS	4.0	15.2	19.75±0.27	0.021±0.001
HyperDeepEns (RS)	4.2	28.7	20.11±0.18	0.019±0.003
MH-NES	<b>2.7</b>	<b>5.0</b>	<b>19.65±0.09</b>	<b>0.017±0.001</b>

While more efficient alternatives have been proposed for constructing ensembles, such as BatchEnsembles (Wen et al., 2020) and Snapshot Ensembles (Huang et al., 2017; Loshchilov & Hutter, 2017), they do not typically outperform deep ensembles. Multi-headed networks are another means to build efficient ensembles which uses a single shared backbone with multiple prediction heads (Lee et al., 2015; Lan et al., 2018). While multi-headed models have been studied before, to the best of our knowledge, there is no prior work that studies the performance impact of searching head architectures. In this work, we employ neural architecture search (NAS) (Elsken et al., 2019) to search for the heads’ architecture in the multi-headed models. Unlike Deep Ensembles, multi-headed models are trained end-to-end, which allows one-shot NAS methods (Bender et al., 2018; Liu et al., 2019) to optimize an ensemble objective.

The key contributions of this paper are as follows:

- We extend the standard NAS search space to sample prediction heads in multi-headed ensembles.
- We evaluate three one-shot NAS methods and demonstrate comparable or better performance to other ensemble search methods such as NES and HyperDeepEns, with significantly less search costs (see Table 1).
- We provide an analysis and shed some light on potential factors affecting the performance of one-shot NAS methods in the multi-headed setting, particularly for larger ensemble sizes.

### 1.1. Related Work

**Ensemble** methods (Hansen & Salamon, 1990; Dietterich, 2000) are extensively used in machine learning research, as both a means to boost performance, but also for better calibration and uncertainty estimates of deep neural networks (Lakshminarayanan et al., 2016). Diversity in the

predictions made by the ensemble base learners is believed to be crucial in order to obtain good ensembles. Some methods induce diversity by training with specialized losses (Lee et al., 2015; Zhou et al., 2018), building ensemble members with different topologies (Zaidi et al., 2020) or different training hyperparameters (Wenzel et al., 2021) to improve ensemble performance. Multi-head networks trained under a unified objective can produce robust ensembles too, by utilizing diversity encouraging specialized losses (Lee et al., 2015), co-distillation (Lan et al., 2018) or both (Dvornik et al., 2019).

**Neural Architecture Search (NAS)** aims to find an optimal neural network architecture that optimizes some objective (e.g. validation loss) (Elsken et al., 2019). Early methods, such as reinforcement learning (Zoph & Le, 2016) and evolutionary algorithms (Real et al., 2019), but also more recent efficient methods (Bender et al., 2018; Elsken et al., 2017; Cai et al., 2019; Liu et al., 2019; Xu et al., 2019; Chen et al., 2021) have shown that NAS can find architectures that can surpass hand-crafted ones. Recently, NAS has also been applied to ensemble learning. Neural Ensemble Search (NES) (Zaidi et al., 2020) builds a pool of strong independent models with different architectures and applies ensemble selection to create the ensembles. On the other hand, AdaNAS (Macko et al., 2019) iteratively adds base learners to an ensemble to improve the ensemble performance. While NES and AdaNAS construct ensembles by independently training the base learners, which is computationally expensive, we focus on multi-headed ensembles that are cheaper to construct and allow us to leverage efficient NAS methods to design their topology.

## 2. Background

### 2.1. Multi-headed Ensembles

Figure 1 depicts an overview of a multi-headed ensemble for a classification task. The network can be divided into two sections: a *lower block* shared by all the heads and a *multi-head block* consisting of  $M$  heads. Each head may have similar or different architectures.

Given a classification task, let  $\mathcal{D}_{train} = \{(x_n, y_n) : n = 1, \dots, N\}$  be the training set, where  $x_n \in \mathbb{R}^D$  is the  $D$ -dimensional input and  $y_n \in \{1, \dots, C\}$  is the label, assumed to be one of the  $C$  classes. We denote a base learner, in our case neural networks, by  $f_\theta$ , where  $\theta$  represents the network parameters. A network takes the input  $x_i$  and outputs a vector of probabilistic class posteriors over the classes as  $f_\theta(x) \in \mathbb{R}^C$ . We construct an ensemble  $F$  of  $M$  members  $f_{\theta_1}, \dots, f_{\theta_M}$  by averaging the output of the networks.

Let  $\ell$  be the cost function. We use an *ensemble-aware loss function*, which includes the ensemble loss term, in addition to the individual head losses to improve the model

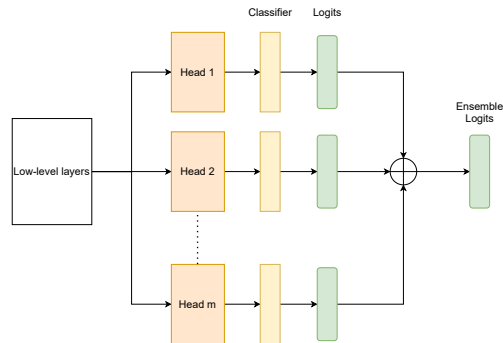


Figure 1. A multi-head ensemble for a CNN classifier

performance:

$$\mathcal{L}_{train} = \sum_{i=1}^M \ell(f_i(x), y) + \ell(F(x), y). \quad (1)$$

### 2.2. Differentiable Architecture Search

Differentiable architecture search (DARTS) (Liu et al., 2019) relaxes the discrete architecture space by assigning continuous architectural parameters to every operation choice in that space. Typically, the search is done for a cell (e.g., convolutional or recurrent) which is stacked in a repetitive manner to form the full network. The cell itself is represented as a directed acyclic graph (DAG) with an ordered sequence of  $N$  nodes. Every node  $x^{(i)}$  denotes a latent representation and each edge  $(i, j)$  is associated with an operation  $o^{(i,j)} \in \mathcal{O}$  that transforms  $x^{(i)}$ , where  $\mathcal{O}$  is a pre-defined space of operations.

The goal is to choose one operation from  $\mathcal{O}$  to connect each pair of nodes. DARTS continuously relaxes these discrete choices by creating a convex combination of the operations in  $\mathcal{O}$  using the mixed operation  $\bar{o}$ :

$$\bar{o}^{(i,j)}(x) = \sum_{o \in \mathcal{O}} \frac{\exp(\alpha_o^{(i,j)})}{\sum_{o' \in \mathcal{O}} \exp(\alpha_{o'}^{(i,j)})} o(x), \quad (2)$$

where  $\alpha_o^{(i,j)}$  is the operation mixing weight. Thus, the problem of searching for the cell architecture boils down to learning a set of continuous variables  $\alpha = \{\alpha^{(i,j)}\}$ . This allows to formulate the NAS problem as a bi-level optimization problem:

$$\min_{\alpha} \mathcal{L}_{val}(w^*(\alpha), \alpha) \quad \text{s.t.} \quad w^*(\alpha) = \arg \min_w \mathcal{L}_{train}(w, \alpha).$$

After the search, the final architecture is derived by retaining the top- $k$  strongest operations from all the previous nodes.

In our work, we use three efficient variations of DARTS, namely:

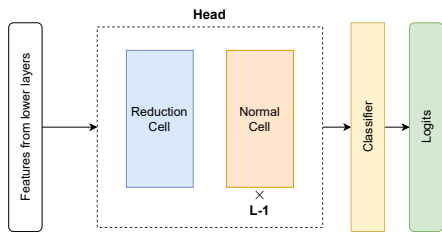


Figure 2. Single-headed network from the search space

**PC-DARTS** (Xu et al., 2019) introduces partial channel connections, where only a portion of channels are sent to the mixed operation. To overcome the instability induced by sampling, it introduces edge weights,  $\beta$ , explicitly for every edge. The connectivity of an edge is then determined by combining both  $\alpha$  and  $\beta$ .

**DrNAS** (Chen et al., 2021) samples the  $\alpha$  parameters from a parameterized Dirichlet distribution. Since it uses partial channel connections, it introduces a multi-stage scheme that increases the channel fraction while pruning the number of operators at every stage.

**RandomNAS** (Li & Talwalkar, 2020) builds a supernet-work similar to DARTS and simply uses randomly sampled architectures to train the shared weights during every mini-batch iteration. The trained shared weights are then used to evaluate the performance of multiple randomly sampled configurations and select the best performing one as the final architecture.

### 3. NAS for multi-headed models

#### 3.1. Search Space

Similar to the DARTS search space (Liu et al., 2019), we use a cell-based architecture for every head, as shown in Figure 2. A head is made of  $L$  cells, with the first cell as a reduction cell (reducing spatial dimensions) and the remaining  $L - 1$  as normal cells (keeping spatial dimensions). Unlike DARTS, we do not distinguish the configurations between reduction and normal cells. We found that having different configurations for reduction and normal cells had little to no impact in the performance of a multi-headed ensemble and only expanded the architecture space. Hence, in a multi-head ensemble with  $M$  heads, there are  $M \times L$  cells but only  $M$  cell configurations or  $M \times \{\alpha\}$  parameters to optimize.

#### 3.2. Diversity Encouraging Loss for Differentiable Search

During the search phase of DARTS, there is no guarantee that the head architectures learnt via gradient descent would be diverse. To this end, we introduce an additional diversity

term only in the loss function for the architecture weights,  $\mathcal{L}_{val}$ . This ensures that the diversity in the one-shot model predictions originates from the architecture weights and not from the network weights.

We use the Jensen-Shannon Divergence (JSD) to measure the diversity between the individual head predictions and maximize it in the validation objective. Unlike KL divergence, JSD is symmetric and bounded (Lin, 1991), which allows for direct maximization without the loss exploding.

Given  $M$  heads, the diversity-encouraging loss term is:

$$\begin{aligned} \mathcal{L}_{jSD} &= JSD[f_{\theta_1}(x), \dots, f_{\theta_M}(x)] \\ &= \frac{1}{M} \sum_{i=0}^M KL[F(x) \parallel f_{\theta_i}(x)] \\ \mathcal{L}_{val} &= \mathcal{L}_{train} - \lambda_{jSD} \mathcal{L}_{jSD} \end{aligned}$$

where  $\lambda_{jSD}$  is the weight of the JSD loss. Refer to Appendix D for ablations.

## 4. Experiments

### 4.1. Setup

**Backbone architecture.** We use a WideResNet-40-2 (Zagoruyko & Komodakis, 2017) backbone for all the methods. The final block is then replaced with 3 cells from our search space.

**Baselines.** We compare our multi-headed ensemble methods with deep ensembles (Lakshminarayanan et al., 2016) (DeepEns (Sample) and DeepEns (RS)), NES-RS (Zaidi et al., 2020) and hyper-deep ensembles (Wenzel et al., 2021) (HyperDeepEns (RS)) from our search space. *Sample* and *RS* refer to head configurations that are randomly sampled and found by random search, respectively. We ran RS for 25 iterations (function evaluations).

**Evaluation.** We compare the methods on NLL, classification error and ECE (Guo et al., 2017). All results are aggregates of 3 runs. We provide more details on the experimental setup in Appendix A.

### 4.2. Results

Figure 3 shows the performance comparison of the ensemble methods on CIFAR-100 and Tiny-ImageNet for an ensemble size  $M = 3$ . Figure 4 shows a comparison between the computational cost and ensemble performance.

As we can see, almost all multi-headed ensembles (even a randomly sampled configuration) show better uncertainty calibration than a deep ensemble. Compared to NES and DeepEns (RS), the one-shot multi-headed search methods, especially DrNAS and RandomNAS, achieve comparable

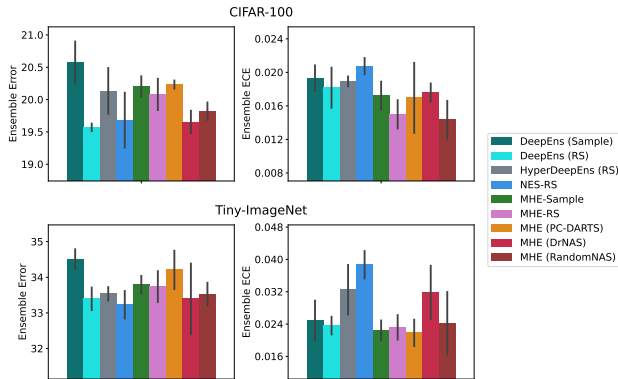


Figure 3. Ensemble performance comparison for  $M = 3$  (mean  $\pm$  std.dev.).  $M$  is the number of ensemble members or prediction heads.

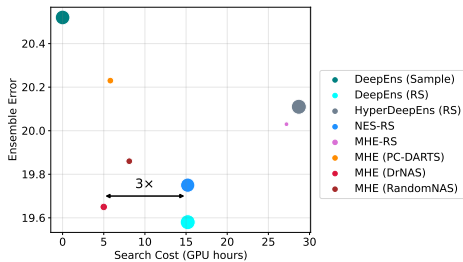


Figure 4. Ensemble error vs search cost (for  $M = 3$ ) on CIFAR-100. Data points closer to the origin are better. Size of each point is proportional to the final ensemble’s model size.

predictive performance, while being  $3\times$  more efficient during search. Among the multi-headed methods, configurations identified by DrNAS perform the best in terms of prediction error. This can be attributed to the natural exploration-exploitation trade-offs in distribution learning. RandomNAS also performed almost as well for small ensembles and for CIFAR-100 identified better configurations than a naive and expensive random search.

Interestingly, for larger ensemble sizes the performance of differentiable methods deteriorates while methods based on random search achieve better performance (Appendix B). We provide more insights into this behavior in the following section.

### 4.3. Analysis of Differentiable Search Methods

Zela et al. (2020) demonstrated that DARTS can often find degenerate architectures depending on the search space at hand. Similarly, we found that PC-DARTS and DrNAS struggled to find architectures better than random search, or even random *samples*, especially for larger ensemble sizes.

A Hessian norm analysis of  $\nabla^2 \mathcal{L}_{valid}$  (Zela et al., 2020) of these methods (Appendix C) indicated that this was not

attributed to architecture over-fitting in the one-shot model: both PC-DARTS and DrNAS were able to find well performing configurations for a single network from the search space, though not for the multi-headed space.

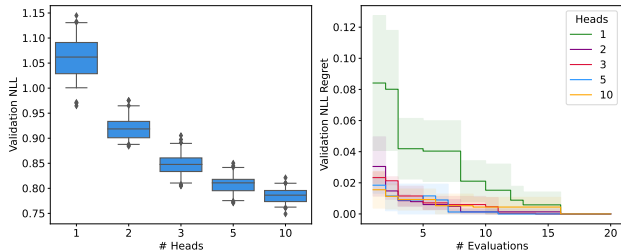


Figure 5. (left) Distribution of validation NLL from 60 samples across 3 random search runs. (right) Validation NLL Regret from optimal configuration of random search (mean  $\pm$  std.dev.). All results on CIFAR-100.

To this end, we analyze the search space by randomly sampling configurations for different ensemble sizes. Figure 5 shows that while there is a clear improvement in the mean ensemble performance when scaling up the number of heads, the variance in the ensemble performance decreases significantly. The validation regret curves flatten out with more heads, indicating that diminishing returns can be expected for running a more complex search method for  $M > 5$  (while increasing computational demands). A similar behaviour can be observed on Tiny-ImageNet in Appendix C.

Furthermore, while most NAS methods previously searched for only 2 cells (Liu et al., 2019; Zhou et al., 2019; Xu et al., 2019; Chen et al., 2021), our search space size increases exponentially with the number of ensemble members/heads. We hypothesize that this, coupled with a low variance regime, leads the DARTS-based methods to get stuck in local optima and find suboptimal configurations. This could also explain why random search traverses the operation search space better and identifies better performing configurations.

## 5. Conclusion and Future Work

In this work, we introduced a method for efficiently searching for the architectures of the individual heads in a multi-headed ensemble. The resulting models are accurate, well calibrated and also more efficient (computationally and memory-wise) compared to deep ensembles and other methods found using inefficient methods that require training every ensemble base learner individually. In the future, we would like to improve the robustness of differentiable NAS methods applied to ensemble learning.

## References

- Bender, G., Kindermans, P.-J., Zoph, B., Vasudevan, V., and Le, Q. Understanding and simplifying one-shot architecture search. In *International Conference on Machine Learning*, pp. 550–559. PMLR, 2018.
- Cai, H., Zhu, L., and Han, S. Proxylessnas: Direct neural architecture search on target task and hardware, 2019.
- Chen, X., Wang, R., Cheng, M., Tang, X., and Hsieh, C.-J. Dr{nas}: Dirichlet neural architecture search. In *International Conference on Learning Representations*, 2021. URL <https://openreview.net/forum?id=9FWas6YbmB3>.
- Dietterich, T. G. Ensemble methods in machine learning. In *International workshop on multiple classifier systems*, pp. 1–15. Springer, 2000.
- Dvornik, N., Schmid, C., and Mairal, J. Diversity with cooperation: Ensemble methods for few-shot classification. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 3723–3731, 2019.
- Elsken, T., Metzen, J.-H., and Hutter, F. Simple and efficient architecture search for convolutional neural networks, 2017.
- Elsken, T., Metzen, J. H., Hutter, F., et al. Neural architecture search: A survey. *J. Mach. Learn. Res.*, 20(55):1–21, 2019.
- Falkner, S., Klein, A., and Hutter, F. Bohb: Robust and efficient hyperparameter optimization at scale, 2018.
- Fort, S., Hu, H., and Lakshminarayanan, B. Deep ensembles: A loss landscape perspective. *arXiv preprint arXiv:1912.02757*, 2019.
- Guo, C., Pleiss, G., Sun, Y., and Weinberger, K. Q. On calibration of modern neural networks. In *International Conference on Machine Learning*, pp. 1321–1330. PMLR, 2017.
- Gustafsson, F. K., Danelljan, M., and Schön, T. B. Evaluating scalable bayesian deep learning methods for robust computer vision, 2020.
- Hansen, L. K. and Salamon, P. Neural network ensembles. *IEEE transactions on pattern analysis and machine intelligence*, 12(10):993–1001, 1990.
- Huang, G., Li, Y., Pleiss, G., Liu, Z., Hopcroft, J. E., and Weinberger, K. Q. Snapshot ensembles: Train 1, get m for free, 2017.
- Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization, 2017.
- Lakshminarayanan, B., Pritzel, A., and Blundell, C. Simple and scalable predictive uncertainty estimation using deep ensembles. *arXiv preprint arXiv:1612.01474*, 2016.
- Lan, X., Zhu, X., and Gong, S. Knowledge distillation by on-the-fly native ensemble. *arXiv preprint arXiv:1806.04606*, 2018.
- Lee, S., Purushwalkam, S., Cogswell, M., Crandall, D., and Batra, D. Why m heads are better than one: Training a diverse ensemble of deep networks. *arXiv preprint arXiv:1511.06314*, 2015.
- Li, L. and Talwalkar, A. Random search and reproducibility for neural architecture search. In *Uncertainty in Artificial Intelligence*, pp. 367–377. PMLR, 2020.
- Lin, J. Divergence measures based on the shannon entropy. *IEEE Transactions on Information theory*, 37(1):145–151, 1991.
- Liu, H., Simonyan, K., and Yang, Y. Darts: Differentiable architecture search, 2019.
- Loshchilov, I. and Hutter, F. Sgdr: Stochastic gradient descent with warm restarts, 2017.
- Macko, V., Weill, C., Mazzawi, H., and Gonzalvo, J. Improving neural architecture search image classifiers via ensemble learning, 2019.
- Ovadia, Y., Fertig, E., Ren, J., Nado, Z., Sculley, D., Nowozin, S., Dillon, J. V., Lakshminarayanan, B., and Snoek, J. Can you trust your model’s uncertainty? evaluating predictive uncertainty under dataset shift, 2019.
- Real, E., Aggarwal, A., Huang, Y., and Le, Q. V. Regularized evolution for image classifier architecture search. In *Proceedings of the aaai conference on artificial intelligence*, volume 33, pp. 4780–4789, 2019.
- Van der Maaten, L. and Hinton, G. Visualizing data using t-sne. *Journal of machine learning research*, 9(11), 2008.
- Wen, Y., Tran, D., and Ba, J. Batchensemble: An alternative approach to efficient ensemble and lifelong learning, 2020.
- Wenzel, F., Snoek, J., Tran, D., and Jenatton, R. Hyperparameter ensembles for robustness and uncertainty quantification, 2021.
- Xu, Y., Xie, L., Zhang, X., Chen, X., Qi, G.-J., Tian, Q., and Xiong, H. Pc-darts: Partial channel connections for memory-efficient architecture search. *arXiv preprint arXiv:1907.05737*, 2019.
- Zagoruyko, S. and Komodakis, N. Wide residual networks, 2017.
- Zaidi, S., Zela, A., Elsken, T., Holmes, C., Hutter, F., and Teh, Y. W. Neural ensemble search for performant and calibrated predictions, 2020.
- Zela, A., Elsken, T., Saikia, T., Marrakchi, Y., Brox, T., and Hutter, F. Understanding and robustifying differentiable architecture search. In *International Conference on Learning Representations*, 2020. URL <https://openreview.net/forum?id=HlgDNyrKDS>.
- Zhou, H., Yang, M., Wang, J., and Pan, W. Bayesnas: A bayesian approach for neural architecture search, 2019.
- Zhou, T., Wang, S., and Biles, J. A. Diverse ensemble evolution: Curriculum data-model marriage. In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018. URL <https://proceedings.neurips.cc/paper/2018/file/3070e6addcd702cb58de5d7897bfdae1-Paper.pdf>.
- Zoph, B. and Le, Q. V. Neural architecture search with reinforcement learning. *arXiv preprint arXiv:1611.01578*, 2016.

## A. Experiment Details

The models were implemented with a WideResNet-40-2 (Zagoruyko & Komodakis, 2017) backbone. The final block of the WideResNet is replaced with heads from the search space. All algorithms were implemented in PyTorch 1.7.1 and run on NVIDIA RTX 2080 GPUs. All the final evaluation networks were trained for 100 epochs with mini-batch size of 128 using SGD with initial learning rate  $\eta = 0.1$ , momentum 0.9 and weight decay  $3 \times 10^{-4}$ . The learning rate was decayed by a cosine annealing schedule (Loshchilov & Hutter, 2017) towards 0.

The one-shot model of the differentiable search methods is trained for 50 epochs with a batch size of 64 using SGD with the same optimizer settings as the final evaluation for the network parameters. The architecture parameters were initialized uniformly around zero. We used Adam optimizer (Kingma & Ba, 2017) with initial learning rate  $\eta = 3 \times 10^{-4}$ ,  $\beta_1 = 0.5$ ,  $\beta_2 = 0.999$  and weight decay  $10^{-3}$ . Similar to (Liu et al., 2019), learnable affine parameters were disabled during search to avoid rescaling the outputs of the candidate operations. In PC-DARTS, the partial channel parameter  $K$  was set to 4. For DrNAS, we use a two stage progressive learning approach, 25 iterations each. In the first stage, the partial channel parameter  $K = 4$  and all 7 operations are used. For the second stage, half the candidate operations are pruned (i.e., 4 remain) and the network is widened by increasing  $K$  to 2. To avoid bias towards non-parametric operations, we warmstart the one-shot model by not updating the architecture parameters for the first 15 epochs for PC-DARTS and 10 epochs for each stage in DrNAS, following the original implementation.

All experiments are an aggregation of 3 independent runs and the results show the mean and standard deviation, unless specified otherwise. For search methods, an independent run includes one search and one evaluation phase.

### A.1. Baselines

- **DeepEns (Sample)** - Deep Ensembles (Lakshminarayanan et al., 2016) train  $M$  different initializations of a randomly sampled head ( $M = 1$ ) from the search space.
- **DeepEns (RS)** is a deep ensemble built using a the optimal configuration from random search for  $M = 1$ .
- **NES-RS** - The ensemble pool is constructed by sampling single-headed networks from our search space. Then, we run forward selection on the pool to select the ensemble members based on the validation set. This approach can be seen as an adaptation of Neural Ensemble Search (Zaidi et al., 2020) to just the last block. We sampled 25 single-headed configurations to build the ensemble pool.

- **HyperDeepEns (RS)** - Hyperparameter Ensembles (Wenzel et al., 2021) use the same networks as *DeepEns (RS)* but train them with two varying hyperparameters: label smoothing and weight decay along with random initializations to build the ensemble pool. Forward selection is used to select the final ensemble members. We sampled 25 configurations to build the ensemble pool.
- **MHE-Sample** is a random sample from the multi-head search space.
- **MHE-RS** is the final configuration from random search. We randomly sample 25 multi-headed configurations from the search space and choose the best configuration based on the validation NLL.

### A.2. Hyperparameter optimization of search parameters

Since this is a fairly new domain for differentiable search methods, we optimize the hyperparameters of the search phase using BOHB (Falkner et al., 2018), a multi-fidelity Bayesian optimization method that uses an efficient surrogate model to search for good hyperparameter configurations.

The hyperparameters optimized were: network learning rate, network weight decay, architecture learning rate, architecture weight decay, backbone layers and width. We included the number of backbone layers and widening factor to search for a proxy one-shot model that can search faster without compromising on the optimal configuration performance. The incumbent configuration from BOHB was later used to run PC-DARTS and DrNAS for all multi-headed settings.

## B. Additional Results

Table 2. Performance and computation cost of different search methods for ensemble size  $M = 3$  on CIFAR-100.

Method	# Params (M)	Search (GPU hours)	Error	ECE
DeepEns (Sample)	4.1		20.52 $\pm$ 0.12	0.019 $\pm$ 0.001
DeepEns (RS)	4.2	15.2	19.58 $\pm$ 0.03	0.018 $\pm$ 0.001
HyperDeepEns (RS)	4.2	28.7	20.11 $\pm$ 0.18	0.019 $\pm$ 0.003
NES-RS	4.0	15.2	19.75 $\pm$ 0.27	0.021 $\pm$ 0.001
MHE-RS	2.4	27.2	20.03 $\pm$ 0.12	0.015 $\pm$ 0.001
MHE (PC-DARTS)	2.6	5.8	20.23 $\pm$ 0.04	0.017 $\pm$ 0.002
MHE (DrNAS)	2.7	5.0	19.65 $\pm$ 0.09	0.017 $\pm$ 0.001
MHE (RandomNAS)	2.6	8.1	19.86 $\pm$ 0.08	0.015 $\pm$ 0.001

### B.1. Ensemble Performance for larger ensembles

On larger ensembles, the multi-headed methods see significant improvements in ensemble performance and calibration, as shown in Figure 6. However, the configurations

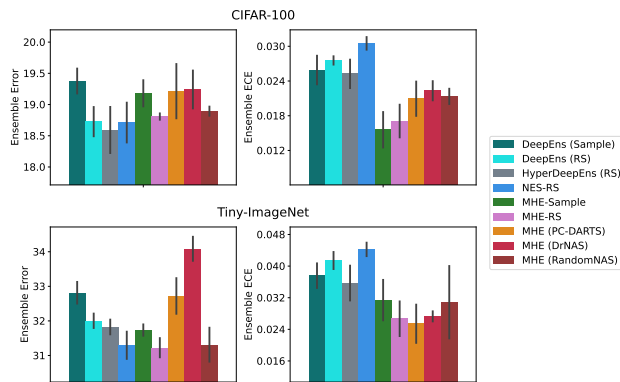


Figure 6. Ensemble performance comparison for  $M = 5$  (mean  $\pm$  std.dev.)

identified by DARTS-based methods, PC-DARTS and DrNAS, deteriorate and perform only as good as a random sample from the search space on CIFAR-100 and even worse on Tiny-ImageNet. The sub-optimal configurations can be attributed to the low variance regime and the increased search space size, which makes them get stuck in a local optima. RandomNAS, the one-shot search method that is based on randomly sampled architectures, is more robust and achieves the best ensemble error for  $M = 5$ .

## B.2. Ensemble Performance under Dataset Shift

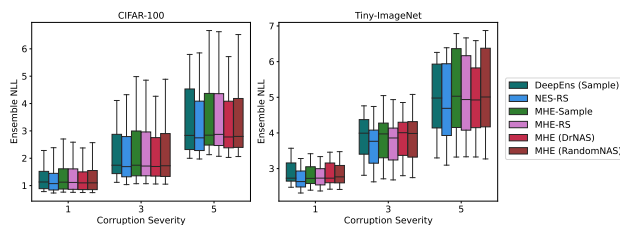


Figure 7. Ensemble performance comparison for  $M = 3$  across different shift severities

The results shown in Figure 7 shows how the ensembles perform on shifted data. The multi-headed search methods mostly outperform the deep ensemble baseline and are even on par with NES on CIFAR-100.

## C. Analysis of DARTS methods

Figure 8 shows the performance of both gradient-based search methods PC-DARTS and DrNAS using the incumbent configuration. Following RobustDARTS (Zela et al., 2020), we plot the dominant eigenvalue of the Hessian of validation loss ( $\nabla^2 \mathcal{L}_{valid}$ ), which serves as a proxy for sharpness, for each search epoch and the corresponding test NLL for 1, 3 and 5 heads.

Table 3. Performance of different search methods for one head on CIFAR-100 dataset. WideResNet block is the baseline block used in the original network.

Method	NLL	Error
WideResNet block	$1.089 \pm 0.006$	$24.41 \pm 0.08$
Random Sample	$0.983 \pm 0.020$	$24.12 \pm 0.21$
Random Search	$0.927 \pm 0.015$	<b><math>23.17 \pm 0.08</math></b>
PC-DARTS	$0.923 \pm 0.008$	<b><math>23.20 \pm 0.10</math></b>
DrNAS	<b><math>0.908 \pm 0.007</math></b>	$23.61 \pm 0.21$

Both PC-DARTS and DrNAS found good single-head configurations that performed at least as well as random search, as shown in Table 3. The DARTS search space is so tuned that random search can offer comparable performance to differentiable methods (Li & Talwalkar, 2020). If we include early stopping based on the dominant eigenvalue like DARTS-ES (Zela et al., 2020), PC-DARTS produces an even better configuration. However, for more heads, the differentiable methods fail to offer much more than a random sample on in-distribution performance. The Hessian norm indicates that this is not a result of architecture over-fitting in the one-shot model.

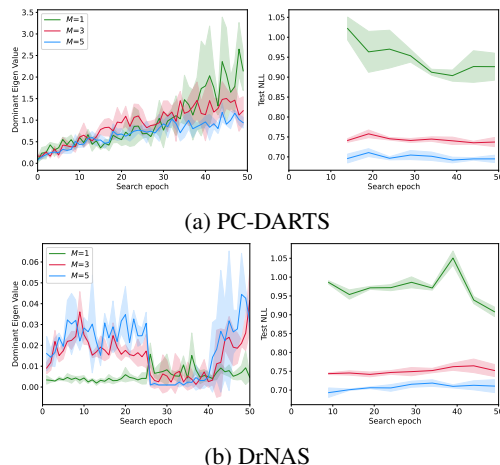


Figure 8. (left) Trajectory of the Hessian norm i.e., dominant Eigenvalue of  $\nabla^2 \mathcal{L}_{valid}$  for PC-DARTS and DrNAS. (right) Test NLL of the optimal architectures by the search methods. All experiments conducted on CIFAR-100.

We build a t-SNE (Van der Maaten & Hinton, 2008) visualization of the head architectures, as shown in Figure 10 to visualize the validation NLL across the architecture search space. We convert the architecture into a vector of edges (operators) for all heads and visualize them using t-SNE with hamming distance as the distance measure. The assumption is that cells that share similar edges would be closer to each other than cells with different operators and this should translate into their performance.

As we can see in Figure 10, the architecture space when  $M = 1$  is defined with bright and dark subspaces, indi-

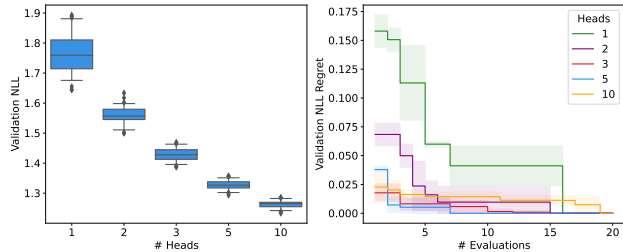


Figure 9. (left) Distribution of validation NLL from 60 samples across 3 random search runs. (right) Validation NLL Regret from optimal configuration of random search (mean  $\pm$  std.dev.). All results on Tiny-ImageNet.

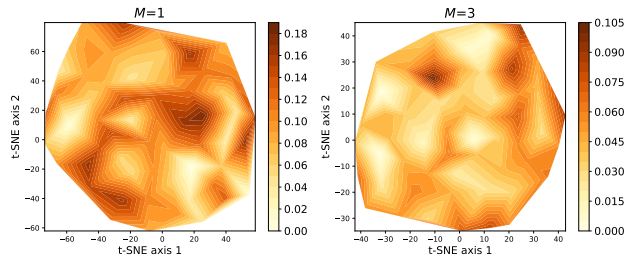


Figure 10. t-SNE visualization of 60 different configurations sampled randomly for ensemble sizes 1 and 3 on CIFAR-100. The regions are coloured based on the validation NLL Regret. Lighter colour indicates better performance. The pervasiveness of bright regions for ensembles implies more well-performing regions in the architecture space.

cating the clear distinction in the quality of architectures. Moreover, the well-performing subspaces are fewer and far between, leading to more informative gradients. Even for a small ensemble  $M = 3$ , while there are still the occasional dark subspaces, the bright spaces become more pervasive, highlighting the similar performance levels across most configurations. The validation NLL regret is predominantly less than 0.04-0.06, which is  $3\times$  smaller than the regret in single heads.

## D. Ablation Studies

### D.1. Diversity Term in Architecture Search

Figure 11 compares the performance of PC-DARTS and DrNAS with and without the diversity constraint on the default search configurations, before hyperparameter optimization. In PC-DARTS, it is clear that the additional constraint improves the performance of final configuration. The dominant eigenvalue of  $\nabla^2 \mathcal{L}_{valid}$  is more regulated, indicating a more stable search. The optimal configuration has a better test performance and diversity performance, demonstrating the benefits of using the diversity term during search. There is no clear winner for DrNAS. While the

diversity term regulated the dominant eigenvalues and led to better early configurations, the final configuration did not improve the diversity or accuracy in the predictions over the unconstrained objective.

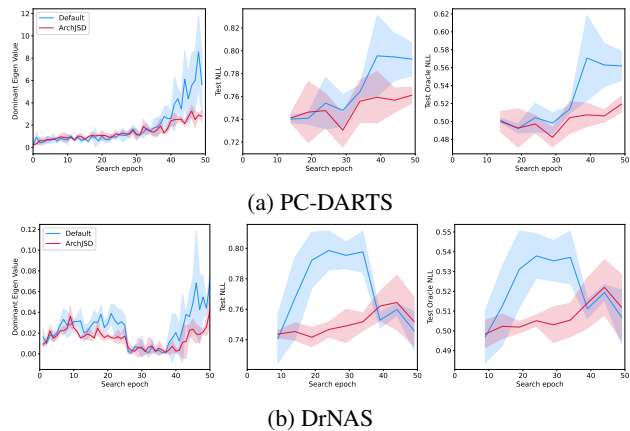


Figure 11. (left) Trajectory of the Hessian norm (dominant eigenvalue of  $\nabla^2 \mathcal{L}_{valid}$ ) during search. (middle) Test NLL of optimal architectures from the search phase. (right) Oracle Ensemble NLL (Zaidi et al., 2020) of the optimal architectures during search. All experiments run on CIFAR-100 with an ensemble size  $M = 3$ . Default configurations were used for the search methods. (mean  $\pm$  std.dev.)

### D.2. Deep Ensembles vs MHE for Similar Architectures

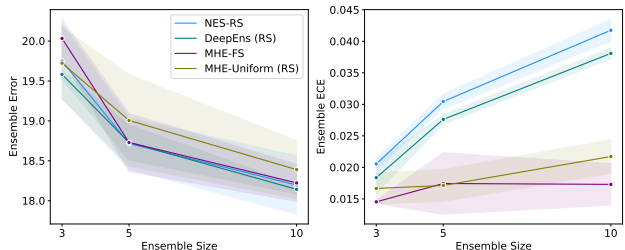


Figure 12. In-distribution performance of independent and multi-headed ensembles on CIFAR-100 dataset. MHE-Uniform (RS) and DeepEns (RS) have same uniform ensemble configuration. NES-RS and MHE-FS have the same ensemble configuration with varying member architectures.

Figure 12 compares the ensemble error and ECE of deep ensembles and multi-headed ensembles on CIFAR-100 for increasing ensemble sizes. Both MHE methods are better calibrated compared to deep ensembles, which is clearly evident for larger values of  $M$ . For  $M = 3$ , MHE with varying configurations does not perform as well as the other methods. A good uniform configuration can perform better than varying configurations for smaller ensembles, even for deep ensembles. The effect of varying architectures shows only for medium to large ensembles, when  $M \geq 5$ .