DIFFERENTIAL NEURAL ENSEMBLE SEARCH WITH DIVERSITY CONTROL

A PREPRINT

P. Babkin, K. Yakovlev, K. Petrushina, O.Bakhteev

ABSTRACT

In our research, we investigate a novel method for sampling deep learning models using a hypernetwork. The hypernetwork is a neural network that controls the diversity of the models by translating a parameter representing ensemble diversity into a neural network architecture. Architectures are obtained in a one-shot manner by perturbing from the base architecture. We evaluate the performance of the proposed algorithm by conducting experiments on the CIFAR-100 datasets, validating method and comparing the resulting ensembles with those sampled by other searching algorithms.

Keywords differential search · neural ensembles · hypernetwork · diversity control

1 Introduction

Nowadays methods of neural architecture search (NAS) are well-explored and proved to be an effective way of creating more efficient neural networks [24, 13, 25, 1]. On the other hand, neural ensemble search (NES) is modern and not as well investigated problem as NAS, although it is well-established that ensembles of deep learning models exhibit superior performance and possess greater robustness compared to individual models [12, 17, 2].

A straightforward approach to constructing neural network ensembles involves generating multiple random initializations of the architecture search algorithm. Ensembles created through this method (DeepEns) are capable of achieving higher predictive capabilities compared to methods for searching single model architectures [9, 5]. Subsequently, the field progressed, and various more efficient approaches to constructing DeepEns emerged, taking into account model diversity and striving to search for them more effectively [3, 10, 19]. However, the sequential training of models is fraught with substantial computational costs, as even a single run of the architecture search process is a computationally intensive task [16].

Contemporary researchers are interested in constructing ensembles of neural networks in a one-shot manner [16, 12]. This approach aims to circumvent the computational overhead associated with sequential model training by generating an ensemble of architectures simultaneously. The premise behind one-shot ensemble generation lies in the efficient exploration of the architecture space, seeking for diverse and well-performing architectures.

As Neural Architecture Search (NAS) methods are more established and well-investigated, while Neural Ensemble Search (NES) represents a more recent yet closely related domain, NES techniques frequently build upon the investigations conducted within the NAS field. Our method is no exception and is grounded in the DARTS [11] methodology.

Our paper investigates a method of sampling deep learning models in a new way that gives compatible results. Our method takes the result of NAS as a base architecture than it samples architectures which are close to the optimal one in terms of shared connections. Architectures differ in terms of λ, idea of method is shown in Fig. 1. Basic architecture is gained with $\lambda = \lambda_1$. Blue ellipses are equidistant surfaces of architectures. Staring with diverse parameter $\lambda = \lambda_2$ resulting architecture performs unacceptable accuracy, so architectures beyond the surface are not included into ensemble. Method can control whether sampled architectures are close enough to the optimal one so they perform good accuracy on the original dataset and are diverse enough so every architecture makes its own contribution to the final answer.

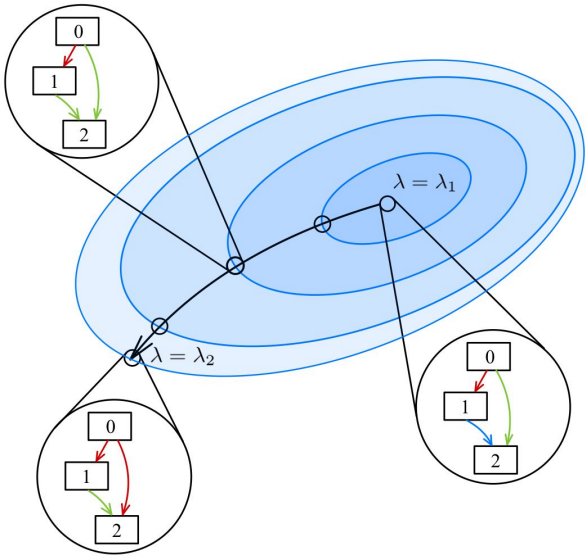

Figure 1: Architecture space with JSd metrics. Blue ellipses are equidistant surfaces. The space is continuous but has varying preferred operations across different architectural levels. Architectures with different operations are pictured separately on the picture.

To sample architectures we use hypernetwork [4]. This network generates parameters for another network, which is called target network. Previously hypernetwoks were intended to control different characteristics such as complexity of architecture [22] or parameters of the target model [18] in several modern papers. In our paper it controls diversity of the target models, so every sampled model differs from previously sampled ones in terms of JSd.

The hypernetwork uses JSd to measure difference between two architectures. Our main idea of sampling different model is to use a regularizer, based on JSd as a source of diversity.

This way we sample deep learning models in one-shot unlike NES for Uncertainty Estimation [23], where every sampled models must be tested and least appropriate architectures are excluded from ensemble. In this way our method is similar to NES via Bayesian Sampling [**?**], but we sample architectures using different approach, posterior distribution for architectures is gained based on the optimal one. To sum up the scheme of our method: find a base architecture using DARTS, sample architectures in one-shot via differentiable algorithm. Inference answer is ensemble of the sampled deep learning models.

We conduct experiments on CIFAR and MNIST datasets to evaluate performance of the proposed method in terms of accuracy and time. Also we compare the performance with state-of-art NES algorithms.

Notation. We use bold lowercase letters $\mathbf{y}$ to denote vectors, and bold uppercase letters $\mathbf{X}$ to denote matrices. In our paper, we address the problem of supervised learning. We assume that the dataset $\mathcal{D} = (\mathbf{X}, \mathbf{y})$ is given, where $\mathbf{X}$ represents the feature matrix and $\mathbf{y}$ represents the target vector. The dataset is divided into training and validation subsets, for which the loss functions $\mathcal{L}_{train}$ and $\mathcal{L}_{val}$ are specified, respectively. In general, these functions depend on the target vector and the predicted outputs of the training model. However, in our paper, we will not explicitly state their dependence on the dataset, assuming it implicitly. Therefore, we will state that the loss functions depend only on the predicted outputs of the model. Moreover, when appropriate, we shall assume that the model outputs, and hence the loss function, depends on the model architecture and its parameters.

2 Problem Statement

2.1 Neural Architecture Search

Let $\mathcal{V} = \{1, \dots, N\}$ be a set of vertices, where N is a number of vertices, and $\mathcal{E} = \{(i, j) \in \mathcal{V} \times \mathcal{V} \mid i < j\}$ a set of edges between them. A set of possible operations in architecture $\mathcal{O}$ usually contains pooling, convolutions, etc. For each edge there is an operation $o \in \mathcal{O}$ that transits information from one node to another. Thus, the task of neural architecture search is reduced to the task of finding operations $o^{(i,j)}$ for each edge (i, j). Considering $\boldsymbol{\alpha}$ as the vector of

parameters indicating the operations within each edge, the NAS problem can be formally written in the following way:

$$\begin{aligned} &\min_{\boldsymbol{\alpha}} \mathcal{L}_{val}(\boldsymbol{w}^*_{\boldsymbol{\alpha}}, \boldsymbol{\alpha}) \\ s.t. \quad &\boldsymbol{w}^*_{\boldsymbol{\alpha}} = \arg\min_{\boldsymbol{w}} \mathcal{L}_{train}(\boldsymbol{w}, \boldsymbol{\alpha}) \end{aligned}. \tag{1}$$

In this problem, the challenge lies in the fact that the architecture space is exponentially large with respect to the number of vertices.

2.2 Neural Ensemble Search

Contrary to the selection of one single architecture in conventional NAS algorithm [11**?**], this paper focuses on the problem of selecting a well-performing neural network ensemble with diverse architectures from the NAS search space, i.e. neural ensemble search (NES).

In order to facilitate our study, several key terms are defined. The architecture of a model, i.e. a set of node operations, is denoted by $\boldsymbol{\alpha}$. For a fixed architecture, optimal parameters are denoted by $\boldsymbol{w}^*_{\boldsymbol{\alpha}}$. The output of an architecture $\boldsymbol{\alpha}$, given its corresponding model parameters $\boldsymbol{w}_{\boldsymbol{\alpha}}$, is denoted by $f(\boldsymbol{w}_{\boldsymbol{\alpha}},\ \boldsymbol{\alpha})$. Finally, the set of architectures included in the ensemble is denoted by $\mathcal{S}$.

Given the ensemble scheme, NES can be formally framed as

$$\begin{aligned} &\min_{S} \mathcal{L}_{val}\left(\frac{1}{|S|}\sum_{\boldsymbol{\alpha}\in S} f(\boldsymbol{w}^*_{\boldsymbol{\alpha}}, \boldsymbol{\alpha})\right) \\ s.t. \quad &\forall \boldsymbol{\alpha} \in S \quad \boldsymbol{w}^*_{\boldsymbol{\alpha}} = \arg\min_{\boldsymbol{w}} \mathcal{L}_{train}(f(\boldsymbol{w}, \boldsymbol{\alpha})) \end{aligned}. \tag{2}$$

In this task, there is an issue of an exponentially large architecture space inherited from NAS. Additionally, there is a problem of an exponentially large number of architecture combinations in the resulting ensemble $\mathcal{S}$. To address these challenges, a method of architecture space smoothing has been devised.

2.3 Architectural Space

NAS algorithms search for optimal architecture. As it was mentioned below, architecture of neural network is a set of operations between nodes. In differential NAS methods architecture is a vector constructed by following rules. For each edge $(i, j) \in \mathcal{E}$, $\boldsymbol{\alpha}^{(i,j)} \in \mathbb{R}^{|\mathcal{O}|}$ is a vector, which assigns impact of each operation [11].

In our paper, we substantially employ the edge-normalization approach [21], which has demonstrated promising results in enhancing architecture search. In this approach, the magnitude of contribution varies not only for operations within edges but also for the edges themselves. Thus, the algorithm modification assists in selecting not only operations within edges but also the presence of edges themselves. This enables the resulting architectures to be more compact and uncluttered. For each vertex $i \in \mathcal{V}$, $\boldsymbol{\beta}^{(i)}$ is a vector that represents the contribution of edges leading to a given node.

Concatenation of all structural parameters vectors $\boldsymbol{\alpha}^{(i,j)}$ and $\boldsymbol{\beta}^{(i)}$ is denoted as $\boldsymbol{\alpha}$. This vector contains all information about smoothed architecture. From this vector, the resulting discrete architecture can be obtained through a straightforward application of one-hot choice.

3 Method

3.1 Shared Connections Regularizer

The proposed method constructs an ensemble, starting from the optimal architecture, which is obtained by solving the optimization problem (1). Subsequently, the method searches for architectures that differ from the initial one by selecting alternative connections. In doing so, the method seeks the most optimal way to replace a certain number of connections. To accomplish this step, we employ a regularizer in our work, which facilitates obtaining the desired architectures during the optimization process.

Transitioning to a more formal description of the method, let $\boldsymbol{\alpha}^*$ denote the optimal architecture obtained by solving the optimization problem (1). To quantify architectural diversity, we define λ, a parameter ranging from 0 to Λ, where Λ is the predefined maximum number of edges in the resulting architectures.

To compute the number of shared edges with the optimal architecture $\boldsymbol{\alpha}^*$, we employ the dot product of the parameter vectors of these architectures. For architecture smoothing, we apply the Gumbel-softmax operation [6]. It incorporates

a temperature parameter t that governs the degree of discretization of the resulting architecture, thus allowing control over the regularizer's stringency on par with the regularizer weight parameter.

Given the parameter λ we reformulate problem (1) in the following way

$$\begin{gathered}\min_{\boldsymbol{\alpha}} \mathcal{L}_{val}(\boldsymbol{w}^*, \boldsymbol{\alpha}) + c(\lambda - \langle \boldsymbol{\alpha}^*, GS(\boldsymbol{\alpha})\rangle)^2 \\ s.t. \quad \boldsymbol{w}^* = \arg\min_{\boldsymbol{w}} \mathcal{L}_{train}(\boldsymbol{w}, \boldsymbol{\alpha})\end{gathered}, \tag{3}$$

where c is weight of the regularizer. In addressing the given task, one can obtain architectures that differ from the original by a certain number of edges, controlled by the diversity parameter λ. At the same time, the discovered architectures will be optimal for the specified number of edges. The latter statement has been validated through computational experimentation.

3.2 Diversity Control via Hypernetwork

In the previous subsection, a method for controlling diversity for two architectures was discussed. However, to construct an ensemble, a means of controlling the diversity of a set of architectures is required. To address this task, we employ the concept of hypernetwork [4]. A hypernetwork in our research is a parametric mapping from $[0, \Lambda]$ to the set of model structural parameters [22]

$$\boldsymbol{\alpha} : [0, \Lambda] \times \mathbb{R}^u \to \mathbb{R}^s,$$

where $\mathbb{R}^u$ is a hypernetwork parametric space and $\mathbb{R}^s$ is space of model structural parameters. In this terms $\boldsymbol{\alpha}$ can be redefined using hypernetwork

$$\boldsymbol{\alpha} = \boldsymbol{\alpha}(\lambda, \boldsymbol{a}),$$

where $\boldsymbol{a}$ is a parameter vector of the hypernetwork. In this paper $\boldsymbol{\alpha}$ is a piecewise-linear function, i.e.

$$\boldsymbol{\alpha}(\lambda, \boldsymbol{a}) = \sum_{k=0}^{K-1} \left(\frac{\lambda - r_k}{r_{k+1} - r_k} \boldsymbol{a_k} + \left(1 - \frac{\lambda - r_k}{r_{k+1} - r_k}\right) \boldsymbol{a_{k+1}} \right) I[\lambda \in [r_k, r_{k+1}]],$$

where $\boldsymbol{a_k}$ and r_k are trainable parameters for each k and K is predefined amount of pivots. The training of the hypernet is a key task for ensemble construction in our work. Let us formulate the training objective as

$$\begin{gathered}\min_{\boldsymbol{a}} \mathbb{E}_{\lambda \sim p(\Lambda)}[\mathcal{L}_{val}(\boldsymbol{w}^*, \boldsymbol{\alpha}(\lambda, \boldsymbol{a})) + c(\lambda - \langle \boldsymbol{\alpha}^*, GS(\boldsymbol{\alpha}(\lambda, \boldsymbol{a}))\rangle)^2] \\ s.t. \quad \boldsymbol{w}^* = \arg\min_{\boldsymbol{w}} \mathbb{E}_{\lambda \sim p(\Lambda)}[\mathcal{L}_{train}(\boldsymbol{w}, \boldsymbol{\alpha}(\lambda, \boldsymbol{a}))]\end{gathered}, \tag{4}$$

where $p(\Lambda)$ is a predefined distribution over the set $[0, \Lambda]$. During optimization process we sample λ from $p(\Lambda)$ and perform optimization steps in two stages. Due to the structure of the regularizer described in the section 3.1, during the iterative optimization of the network, architectures with many overlapping edges will be sampled, thereby accelerating the optimization process and enabling the algorithm to operate more efficiently [15, 20].

During the training of the hypernetwork, it is crucial to strike a balance in the weight of the regularizer. An excessively small weight will prevent the regularizer from serving as a source of diversity, making it impossible to control the diversity of architectures. Conversely, an excessively large weight will inhibit the algorithm's ability to escape local minima induced by the regularizer, hindering its capacity to identify well-performing architectures. The same can be said about the magnitude of discretization, that is, the temperature parameter t in the Gumbel-softmax distribution. An appropriate balance must be maintained for this parameter as well. An overly small value of t would lead to premature discretization, limiting the exploration of the architecture space and potentially trapping the algorithm in sub-optimal solutions. On the other hand, an excessively large value of t would result in slow convergence and inefficient utilization of computational resources, as the search process would remain diffuse and unfocused for an extended period.

3.3 Ensembling Strategy

Summarizing the above, we present an Algorithm 1 to solve the problem (2) using differentiable search with controlled diversity of the models. In the first step of the algorithm, an optimal network is obtained by solving the problem defined in Equation (1). Subsequently, a hypernetwork (4) is trained, where the output from the previous step is used as the optimal architecture. This results in a piecewise-linear hypernetwork capable of generating architectures for the ensemble. The hypernetwork is then applied to generate candidate architectures for the ensemble. In the final stage, n architectures are selected, representing the best-performing ensemble on the validation dataset.

Algorithm 1 EdgeNES

Initialize: $n \in \mathbb{N}, \mathcal{S}' = \varnothing, N \in \mathbb{N}$
$\boldsymbol{\alpha}^* \leftarrow$ result of 1
Train hypernetwork $\boldsymbol{\alpha}$ using 4 and $\boldsymbol{\alpha}^*$
for $i = 1, \ldots, N$ **do**
  Sample $\lambda \sim p(\Lambda)$
  $\mathcal{S} \leftarrow \mathcal{S} \cup \{\boldsymbol{\alpha}(\lambda, \boldsymbol{\alpha}^*)\}$
end for
$\mathcal{S} \leftarrow$ best performing ensemble of size n from $\mathcal{S}'$
Return: $\mathcal{S}$ as a resulting ensemble

We should notice that after selecting the candidate architectures for the ensemble, it is necessary to retrain them from scratch for subsequent validation. The retraining process is much cheaper than the architecture search process. Thus, it will not significantly impact the overall runtime.

Additionally, it is essential to remark on the final stage of the algorithm. To find the optimal ensemble, we propose enumerating combinations of candidate architectures and evaluating their performance on the validation dataset. The number of such combinations will be exponentially large with respect to the ensemble size.

This approach is similar to random ensemble search [**?**], but the key difference lies in the fact that the original search space of DARTS contains $\sim 10^{25}$ architectures, whereas the number of candidate architectures in Algorithm 1 equals N. In other words, we are controllably narrowing the search space.

By controllably restricting the search space, this approach alleviates the computational burden associated with exploring an extremely vast search space, which can be prohibitively expensive. Moreover, the candidate architectures for the ensemble are known to perform well on the validation dataset and exhibit differences in terms of their distinct edges, which provides diversity in their predictions.

As a result, we obtain an ensemble of neural networks, each of which show high scores on the validation data. The diversity of architectures lies in the different connections employed in the selected architectures, which ensures a difference in the responses [16].

4 Computational Experiment

In this study, we conduct experiments on the CIFAR-100 dataset [8] using a convolutional neural network (CNN) [14, 25] model to search ensemble in the DARTS [11] search space. Specifically, we utilize a one-layer architecture with six nodes and seven possible operations for each edge. Our optimizer is Adam [7] for both stages of optimization, and we employ a cosine annealing learning rate scheduler. The loss criterion utilized in this study is CrossEntropyLoss.

In our work, instead of setting constant parameters, we employ an annealing technique. Specifically, we linearly increase the weight of the regularizer c and linearly decrease the temperature t during the optimization process, i.e.

$$c_i = c_{start} + (c_{end} - c_{start})\frac{e_{current}}{e_{total}} \quad \text{and} \quad t_i = t_{start} + (t_{end} - t_{start})\frac{e_{current}}{e_{total}},$$

where c_{start}, t_{start} are initial values of the weight and temperature, c_{end}, t_{end} are values in the end of the optimization process, $e_{current}, e_{total}$ are current epoch of process and total amount of epochs respectively. By selecting $c_{start} < c_{end}$ and $t_{start} > t_{end}$, we can obtain architectures that exhibit controlled diversity while simultaneously demonstrating robust performance.

In this setup, we conduct a series of experiments to validate the proposed method, investigate speed of learning and evaluate its efficiency.

4.1 Hypernetwrok Validation

In this experiment, we validate results of hypernetwork to assess the predictive capability of the obtained architectures. We employ DARTS to obtain an optimal architecture, and subsequently, we train the hypernetwork using (4). To evaluate the effectiveness of the gained hypernetwork, we introduce a controlled perturbation by randomly replacing a

certain number of edges in the optimal architecture. This controlled modification allows us to assess the hypernetwork's ability to sample effective architectures that deviate from the optimal configuration. The derived architectures are retrained and evaluated for accuracy performance on a validation dataset.

The results of the seven experimental runs are presented in Table 1, providing an overview of the architectures predictive capabilities. Each cell in the table reports the mean and variance values of accuracy in percents on a validation dataset, enabling a detailed analysis of the prediction accuracy and consistency across different architectural modifications. The top row of the table indicates the number of shared edges with the optimal architecture, facilitating the interpretation of the results in relation to the degree of perturbation introduced.

λ	0	1	2	3
Hypernetwork	92.10	92.10	92.10	92.10
Random Deviation	89.66	92.10 %	92.10	92.10

Table 1: The accuracy performance achieved by the architectures gained from random deviation and from a hypernetwork. The values presented in the table represent the accuracy performance exhibited by the retrained architectures when evaluated on the validation dataset.

The values from the experiment presented in the table allow us to draw several important conclusions. Firstly, the architectures obtained through the use of the hypernet demonstrate statistically significant superiority in accuracy compared to the architectures derived from the random deviation approach. This finding suggests that the hypernet is capable of discovering more optimal architectural configurations tailored to the given task.

Secondly, when examining the architectures obtained through random deviation, an expected trend emerges: as the number of randomly selected edges increases, the dispersion of accuracy also increases, while the mean accuracy decreases. This can be attributed to the fact that random changes in the architecture are more likely to degrade performance than to improve it.

Thirdly, in analyzing the architectures generated by the hypernetwork, a similar pattern can be observed: the further the obtained architecture deviates from the optimal configuration in terms of common connections, the lower its accuracy becomes. This result correlates with the Figure 1. And reinforces the method from a theoretical perspective.

Furthermore, it is noteworthy that the hypernetwork's ability to maintain high performance while exploring new configurations plays a crucial role in creating ensembles of the deep learning models. By navigating the vast search space of possible architectures, the hypernetwork can strike a balance between exploiting the strengths of the original architecture and exploring deviations from it. Consequently, the conducted experiments highlight the advantages of employing a hypernetwork for discovering optimal deviations from the optimal network.

4.2 Convergence Speed

In the following experiment, we compare the performance of the algorithm in terms of accuracy and speed with a baseline. The baseline in our case is the ensembling of DARTS [11]. In other words, multiple runs of the DARTS algorithm are performed, and the obtained architectures are ensembled to produce collective outputs. Despite the simplicity of this method, it serves as a strong baseline in the field of neural network ensembling [].

A comparative plot is constructed in Figure 2, illustrating the relationship between ensembling accuracy and the number of training iterations, which can be equated to training time. According to the graph, the advantage of our proposed method is evident: an efficient ensemble is discovered more rapidly compared to training multiple models individually, corroborating the theoretical estimates presented in Section 3.2.

4.3 Ensembling Results

In the final experiment, we conduct a comparative evaluation of the ultimate predictive accuracy achieved on the test dataset and the computational runtime across the various methods under consideration. The results, tabulated in Table 2, encompass both single-model approaches and ensemble-based techniques.

From the values given in Table 2 it is evident that ensemble-based approaches exhibit the potential to achieve higher predictive accuracy with enhanced robustness, thereby corroborating the efficacy of model ensembling techniques in deep learning paradigms.

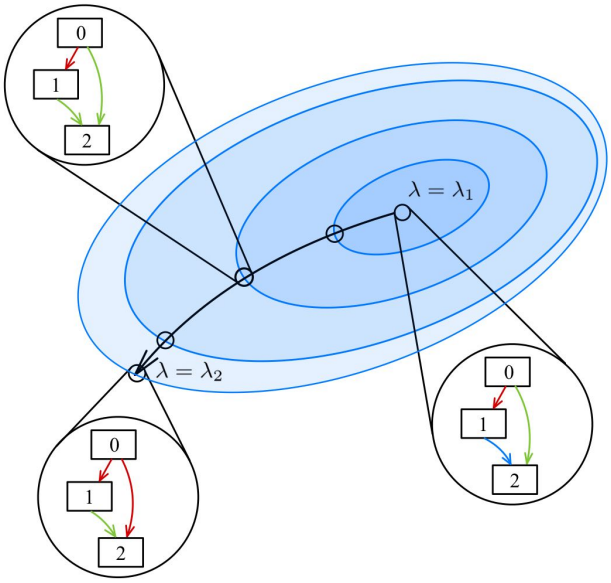

Figure 2: The graphical representation delineates the evolution of predictive accuracy on the test dataset for ensemble configurations, as a function of the training iterations, contrasting the performance trajectories of proposed EdgeNES 1 and the DARTS ensemble approach.

Method	Test Accuracy	Search cost (GPU hours)
DARTS	92.10	92.10
Random	89.66	92.10 %
DARTS-DeepEns	89.66	92.10 %
Random-NES	89.66	92.10 %
RandomD-NES	89.66	92.10 %
EdgeNES	89.66	92.10 %

Table 2: A comparative evaluation of the results obtained from diverse methodological approaches, assessing their predictive accuracy on held-out test data and associated computational runtimes.

Furthermore, the proposed EdgeNES method demonstrates a clear superiority over the other methods included in the analysis. Compared to the ensemble-based DARTS approach, EdgeNES operates with greater computational efficiency without compromising predictive accuracy. Additionally, when contrasted with randomly generated architectures, EdgeNES excels in achieving higher accuracy levels and exhibits greater robustness.

From the empirical findings, it can be inferred that the EdgeNES methodology represents a promising avenue for leveraging the benefits of ensemble-based optimization while mitigating the associated computational overhead. Its ability to strike an optimal balance between predictive performance and computational efficiency renders it a compelling choice for practical applications with stringent resource constraints or real-time performance requirements.

5 Conclusion

In this paper, we proposed a novel method for sampling ensembles of deep learning models with diversity control. Our method utilizes a hypernetwork to generate diverse architectures by perturbing a base architecture in terms of common connections divergence. The diversity of the ensemble is controlled by a penalty term added to the loss function, which encourages the ensemble members to be diverse We conducted extensive experiments on the CIFAR-100 dataset and demonstrated that our method performs compatible results in terms of accuracy. Overall, proposed method shows potential for practical applications in deep learning ensembling.

References

[1] B. Baker, O. Gupta, N. Naik, and R. Raskar. Designing neural network architectures using reinforcement learning. *arXiv preprint arXiv:1611.02167*, 2016.

[2] T. G. Dietterich. Ensemble methods in machine learning. In *International workshop on multiple classifier systems*, pages 1–15. Springer, 2000.

[3] S. Fort, H. Hu, and B. Lakshminarayanan. Deep ensembles: A loss landscape perspective. *arXiv preprint arXiv:1912.02757*, 2019.

[4] D. Ha, A. Dai, and Q. V. Le. Hypernetworks. *arXiv preprint arXiv:1609.09106*, 2016.

[5] L. K. Hansen and P. Salamon. Neural network ensembles. *IEEE Trans. Pattern Anal. Mach. Intell.*, 12:993–1001, 1990.

[6] E. Jang, S. Gu, and B. Poole. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*, 2016.

[7] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *Proc. ICLR*, 2015.

[8] A. Krizhevsky, G. Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[9] A. Krogh and J. Vedelsby. Neural network ensembles, cross validation, and active learning. In *Proc. NIPS*, pages 231–238, 1994.

[10] B. Lakshminarayanan, A. Pritzel, and C. Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. *Advances in neural information processing systems*, 30, 2017.

[11] H. Liu, K. Simonyan, and Y. Yang. Darts: Differentiable architecture search. *arXiv preprint arXiv:1806.09055*, 2018.

[12] A. R. Narayanan, A. Zela, T. Saikia, T. Brox, and F. Hutter. Multi-headed neural ensemble search. *arXiv preprint arXiv:2107.04369*, 2021.

[13] N. Nayman, A. Noy, T. Ridnik, I. Friedman, R. Jin, and L. Zelnik-Manor. XNAS: Neural architecture search with expert advice. In *Proc. NeurIPS*, pages 1975–1985, 2019.

[14] K. O'shea and R. Nash. An introduction to convolutional neural networks. *arXiv preprint arXiv:1511.08458*, 2015.

[15] H. Pham, M. Guan, B. Zoph, Q. Le, and J. Dean. Efficient neural architecture search via parameters sharing. In *International conference on machine learning*, pages 4095–4104. PMLR, 2018.

[16] Y. Shu, Y. Chen, Z. Dai, and B. K. H. Low. Neural ensemble search via bayesian sampling. In *Uncertainty in Artificial Intelligence*, pages 1803–1812. PMLR, 2022.

[17] T. Strauss, M. Hanselmann, A. Junginger, and H. Ulmer. Ensemble methods as a defense to adversarial perturbations against deep neural networks. *arXiv preprint arXiv:1709.03423*, 2017.

[18] J. Von Oswald, C. Henning, B. F. Grewe, and J. Sacramento. Continual learning with hypernetworks. *arXiv preprint arXiv:1906.00695*, 2019.

[19] F. Wenzel, J. Snoek, D. Tran, and R. Jenatton. Hyperparameter ensembles for robustness and uncertainty quantification. *Advances in Neural Information Processing Systems*, 33:6514–6527, 2020.

[20] S. Xie, H. Zheng, C. Liu, and L. Lin. Snas: stochastic neural architecture search. *arXiv preprint arXiv:1812.09926*, 2018.

[21] Y. Xu, L. Xie, X. Zhang, X. Chen, G.-J. Qi, Q. Tian, and H. Xiong. Pc-darts: Partial channel connections for memory-efficient architecture search. *arXiv preprint arXiv:1907.05737*, 2019.

[22] K. Yakovlev, O. Grebenkova, O. Bakhteev, and V. Strijov. Neural architecture search with structure complexity control. EasyChair Preprint no. 7973, EasyChair, 2022.

[23] S. Zaidi, A. Zela, T. Elsken, C. C. Holmes, F. Hutter, and Y. Teh. Neural ensemble search for uncertainty estimation and dataset shift. *Advances in Neural Information Processing Systems*, 34:7898–7911, 2021.

[24] B. Zoph and Q. V. Le. Neural architecture search with reinforcement learning. *arXiv preprint arXiv:1611.01578*, 2016.

[25] B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le. Learning transferable architectures for scalable image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8697–8710, 2018.