Research Article

Alexandr Udeneev, Petr Babkin, and Oleg Bahteev

Surrogate assisted diversity estimation in NES

https://doi.org/10.1515/sample-YYYY-XXXX
Received Month DD, YYYY; revised Month DD, YYYY; accepted Month DD, YYYY

Abstract: The automated search for optimal neural network architectures (NAS) is a challenging computational problem, and Neural Ensemble Search (NES) is even more complex. In this work, we propose a surrogate-based approach for ensebmle creation. Neural architectures are represented as graphs, and their predictions on a dataset serve as training data for the surrogate function. Using this function, we develop an efficient NES framework that enables the selection of diverse and high-performing architectures. The resulting ensemble achieves superior predictive accuracy on CIFAR-10 compared to other one-shot NES methods, demonstrating the effectiveness of our approach.

Keywords: NES, GCN, triplet loss, surrogate function.

1 Introduction

Neural network ensembles often demonstrate better accuracy compared to single models, especially in classification and regression tasks [1, 2]. This fact gives rise to the problem of constructing an efficient ensemble of models (NES) [3]. NES, in turn, relies on Neural Architecture Search (NAS) methods, which are extensively studied and applied to search for individual neural network architectures, such as evolutionary algorithms [4, 5], reinforcement learning [6–8], and Bayesian optimization [9, 10]. Selecting an optimal architecture for even a single model is a challenging task, particularly when considering data-specific constraints and computational limitations [11].

The simplest approach for ensemble construction is the use of DeepEns [12], implemented through DARTS [13]. It involves a random search for several architectures, which are then combined into an ensemble. Despite its simplicity in implementation and hyperparameter tuning, this method is computationally expensive. More sophisticated adaptation techniques are presented in some recent works [3, 14, 15], which are designed to efficiently combine multiple networks into an ensemble.

Our research also adapts ideas from NAS for NES, specifically using a surrogate function. Some modern NAS methods widely use surrogate functions to estimate architecture quality without requiring full model training [16–18]. These functions significantly reduce computational costs, expanding the applicability of such methods. For example, in [16], evolutionary algorithms were proposed in combination with surrogate models for real-time semantic segmentation. In [18], a Surrogate-assisted Multiobjective Evolutionary-based Algorithm (SaMEA) is used for 3D medical image segmentation.

In this work, we propose a method for constructing neural network ensembles using a surrogate function that accounts for both model classification accuracy and architectural diversity. Diversity is crucial because ensembles consisting of similar models often fail to provide a significant performance gain. The surrogate function is used to encode the architecture into a latent space [19], which reflects both the diversity and predictive ability of the architectures. Since a neural network architecture is represented as a graph, using a Graph Neural Network (GNN) [20] as a surrogate function [21] seems natural. To train it to predict model diversity, we use Triplet Loss [22], similar to [19]. We validate this approach on CIFAR-10, demonstrating the effectiveness of the surrogate function for predicting diversity and constructing ensembles. We claim

that ensembles constructed in this manner achieve state-of-the-art accuracy compared to one-shot NES algorithms, such as DeepEns [12].

Main Contributions:

1) We propose a method for encoding the DARTS [13] search space into a representation suitable for training a Graph Neural Network (GNN), where graph nodes correspond to operations within the network.

2) We propose a way for training the surrogate function to predict the diversity of architectures.

3) We adapt surrogate functions for ensemble construction, taking into account both predictive performance and architectural diversity.

2 Problem statement

2.1 Neural Architecture Search

Let us consider a set of nodes $\mathcal{V} = \{x_1, \ldots, x_N\}$, representing the layers of a neural network. Additionally, let $\mathcal{O}$ denote the set of possible operations that can be applied to these nodes (e.g., convolutions or poolings). Furthermore, let $\mathcal{A}$ be the set of feasible architectures, represented as vectors.

Denote $\mathcal{L}_{train}$ and $\mathcal{L}_{val}$ as the training and validation losses, respectively. The NAS problem can then be formulated as the search for an optimal architecture α^* that minimizes $\mathcal{L}_{val}(\alpha^*, \omega^*)$, under the constraint that the weights are obtained by minimizing the training loss:

$$\omega^* = \arg\min_{\omega \in \mathcal{W}} \mathcal{L}_{train}(\alpha^*, \omega)$$

This can be expressed as the following optimization problem:

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

The primary challenge in this optimization lies in the immense search space of possible architectures (e.g., in DARTS [13], it is approximately 10^{25}).

2.2 Neural Ensemble Search

The primary objective of NES is to find an optimal ensemble of neural networks whose architectures lie within the NAS search space.

As before, we denote $\alpha \in \mathcal{A}$ as a network architecture and $\omega(\alpha)$ as its corresponding weights. The action of this network on an input x is denoted by $f_\alpha(x, \omega(\alpha))$. Let $S \subset \mathcal{A}$ be a subset of architectures. Then, the NES problem can be formally described as follows:

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

Thus, in addition to searching over a vast number of architectures, we now also need to find the optimal ensemble composition.

3 Method

Table of Notation

Symbol	Description
$\mathcal{A}$	**The space of architectures**
α	**A specific architecture**
$\omega(\alpha)$	**Network weights corresponding to architecture** α
$f_\alpha(x, \omega(\alpha))$	**The output of the model with architecture** α **on input** x
$\mathcal{V}$	**The set of nodes in a normal or reduction cell**
$\mathcal{O}$	**The set of operations on the nodes**
o	**Operation from** $\mathcal{O}$
N	**Number of models in the dataset**

In this work, we consider the transformation of the architecture space proposed in DARTS [13] for application in Graph Convolutional Networks (GCN) (see Section 3.1). In Section 3.2, we analyze the trade-off between accuracy and model diversity for ensemble construction using the Hellinger distance. In Section 3.3, we present the architecture of the surrogate function and describe its working principle, while Section 3.4 provides a detailed discussion of the ensemble construction method based on this surrogate function.

3.1 The space of architectures

One of the key aspects of this work is the training of a surrogate function. The training data consist of neural network architectures, their prediction logits, and the accuracy on a validation dataset.

The models are constructed using the DARTS search space, which includes the following operations:

- separable convolutions of size 3×3 and 5×5;
- dilated separable convolutions of size 3×3 and 5×5;
- 3×3 max pooling;
- 3×3 average pooling;
- identity;
- zero.

As in the original paper, both normal and reduction cells are utilized; however, in our work these cells are generated randomly. In our approach, each cell is represented as a directed acyclic graph consisting of n nodes and m edges, where each edge corresponds to one of the aforementioned operations. The nodes are interpreted as latent states—i.e., the result of applying the operations (specifically, the average of the sum of the outputs of these operations).

Thus, the architecture space $\mathcal{A}$ is defined as the set of directed acyclic graphs constructed from sequential blocks representing normal and reduction cells.

Note that, unlike in the original DARTS, where a continuous relaxation of the operations is employed, in our approach the operations are selected discretely and their weights are not updated during optimization.

In our work, both normal cells and reduction cells consist of five nodes. From each node, two edges emerge, with each edge being assigned an operation randomly chosen from the DARTS search space.

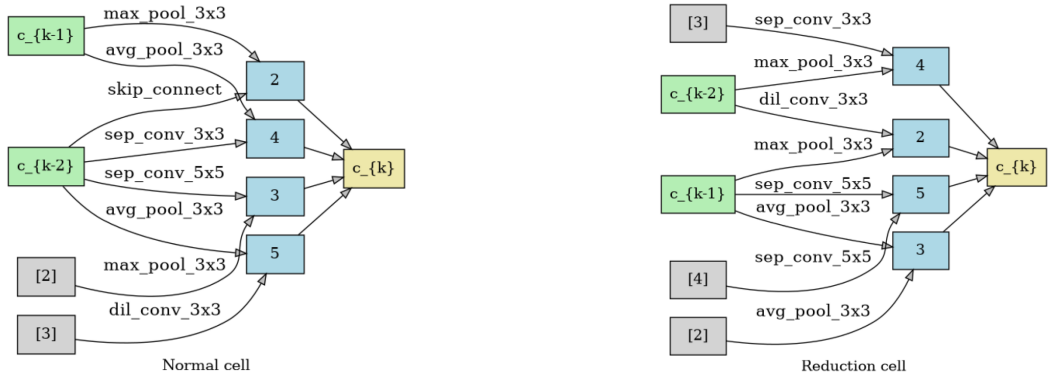

Fig. 1: Example of cells used in the generated models.

The models used for training the surrogate function are built based on two normal cells and one reduction cell. Consequently, the total number of possible architectures is estimated as

$$\left(\binom{2}{2}\cdot\binom{3}{2}\cdot\binom{4}{2}\cdot\binom{5}{2}\cdot 8^8\right)^2 \approx 10^{18},$$

which makes an exhaustive search over all architectures computationally infeasible.

3.2 Performance vs. Diversity Trade-off

3.3 Surrogate Function

In order to construct the ensemble described in Section 3.4, we need to be able to predict both the performance and the similarity of the models. However, due to the enormous number of architectures, directly obtaining these characteristics is unfeasible because of the excessive time requirements.

To address this challenge, we propose to employ a surrogate function — a model that, given an architectural representation, can predict key characteristics of neural network models, such as accuracy or their latent embeddings.

Currently, each architecture in the dataset is represented as a directed acyclic graph (DAG), where the nodes correspond to latent representations and the edges to operations, similar to the format presented in NAS-Bench-201 [23]. For example, the DARTS architectures consist of connected normal and reduction cells (in a 2:1 ratio), as shown in Figure 2.

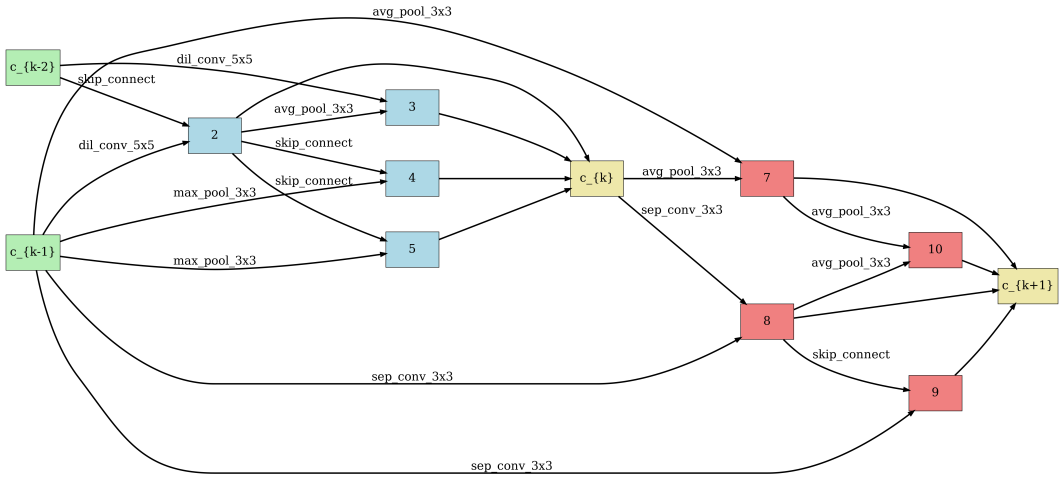

Fig. 2: Combined normal and reduced cells. The red vertices belong to the reduction cell; the green vertices belong to the normal cell.

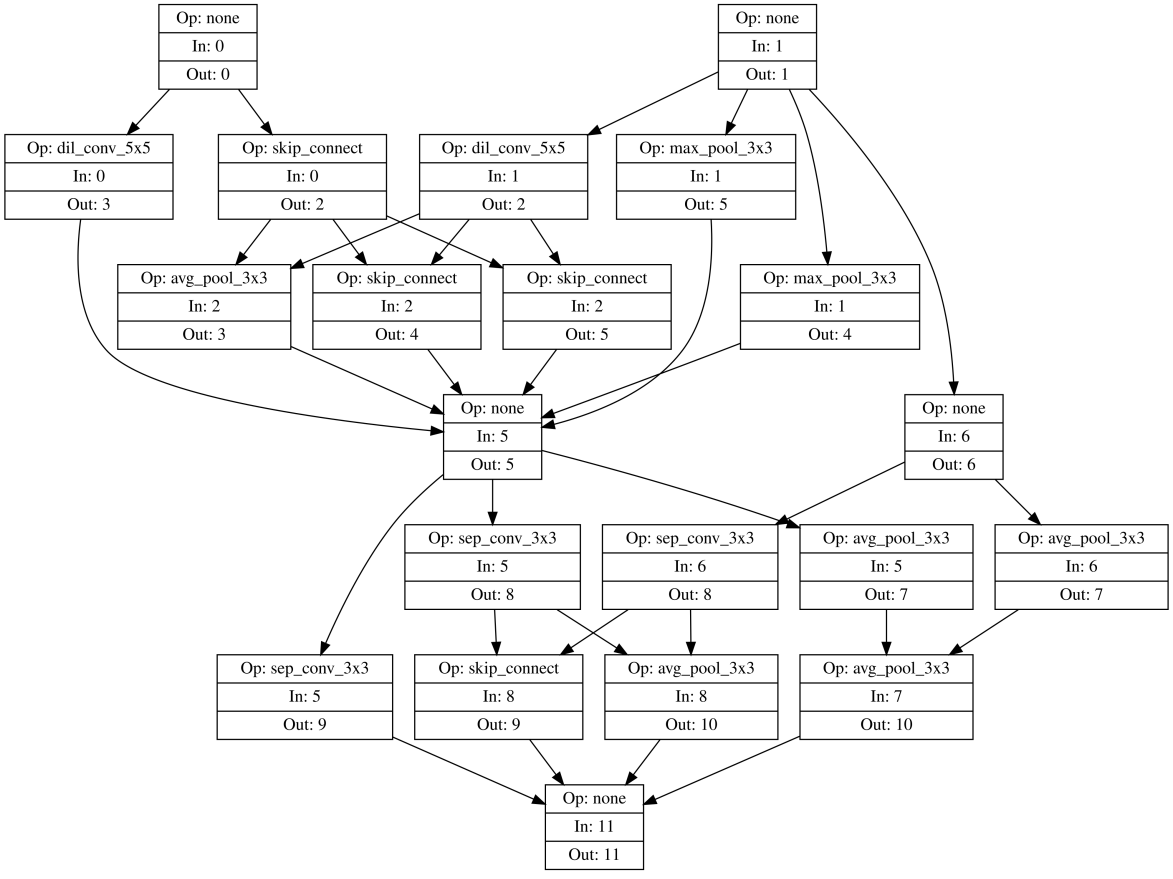

Fig. 3: Conversion of an architecture to the NAS-Bench-101 format.

Subsequently, it is more convenient to adopt an alternative graph representation: the *nodes* represent operations and the *edges* their corresponding latent embeddings. In this way, the architecture is transformed into the NAS-Bench-101 format [24] (see Figure 3).

The conversion is carried out as follows:

1. Each edge of the original graph is transformed into a node in the new graph; the label of this node indicates the operation present on the corresponding edge in the original graph.
2. An oriented edge is drawn from the node with label o_i to the node with label o_j if, in the original graph, the operation o_i acts from a node x to a node y, and the operation o_j acts from node y to a node z (with x, y, and z being arbitrary nodes).

We encode the operations as one-hot vectors. Since our dataset consists of graphs (architectural encodings), we adopt a Graph Convolutional Network (GCN) as the surrogate function, as in [25].

For predicting accuracy, supervised learning can be employed; however, to train the model to predict similarity, we utilize the Triplet Loss [22]. As in [19], Triplet Loss is employed to learn the latent representations of architectures. Similarity between two models can be defined, for instance, by comparing the fraction of identical predictions or by computing the average distance between their output distributions using a suitable metric for multivariate distributions, such as a divergence or distance function in the probability space (e.g., Jensen–Shannon divergence or Hellinger distance). This allows us to construct a similarity matrix of size $N \times N$.

In this article, we construct a similarity matrix based on model responses on the validation dataset. The matrix is constructed as follows:

1. For each of the N models, predictions are computed on a fixed validation dataset consisting of K examples. Denote the predictions of model M_i by the vector
$$\mathbf{y}^{(i)} = \left(y_1^{(i)}, \ldots, y_K^{(i)}\right).$$
2. For every pair of models (M_i, M_j), where $i, j = 1, \ldots, N$, the fraction of matching predictions is computed:
$$s_{ij} = \frac{1}{K} \sum_{k=1}^{K} \mathbb{I}\left(y_k^{(i)} = y_k^{(j)}\right),$$
where $\mathbb{I}$ is the indicator function.
3. The values s_{ij} form the symmetric similarity matrix $\mathbf{S} \in [0,1]^{N \times N}$, with each entry $\mathbf{S}_{ij}$ representing the degree of similarity between models M_i and M_j.
4. To discretize the similarity for the purposes of Triplet Loss, the matrix $\mathbf{S}$ is converted into a discrete matrix $\mathbf{M} \in \{-1, 0, 1\}^{N \times N}$ according to the following rule:
$$\mathbf{M}_{ij} = \begin{cases} 1, & \text{if } s_{ij} > q_p, \\ -1, & \text{if } s_{ij} < q_n, \\ 0, & \text{otherwise,} \end{cases}$$
where q_p and q_n are predetermined thresholds corresponding to the upper and lower quantiles of the distribution of s_{ij} values.

The training algorithm for surrogate similarity function is as follows:

Algorithm 1 Training the surrogate model for architecture diversity

Input: f_{sim}: an untrained surrogate model
$\mathcal{B}$: a set of architectures of pretrained models (in NAS-Bench-101 format)
N: the number of architectures
$\mathbf{M}$: a discrete similarity matrix of size $N \times N$
n: the number of training epochs
`optimizer`: the optimization algorithm
m: the margin parameter for the Triplet Loss
Output: Trained surrogate model f_{sim}

for $i \leftarrow 1$ **to** n **do**
  for $j \leftarrow 1$ **to** N **do**
    `// Sample a positive example`
    $\mathcal{P}_j \leftarrow \{k \mid \mathbf{M}[j,k] = 1\}$
    $k_p \leftarrow \text{UniformSample}(\mathcal{P}_j)$
    `// Sample a negative example`
    $\mathcal{N}_j \leftarrow \{k \mid \mathbf{M}[j,k] = -1\}$
    $k_n \leftarrow \text{UniformSample}(\mathcal{N}_j)$
    `// Compute embeddings`
    $\mathbf{e}_a \leftarrow f_{sim}(\mathcal{B}[j])$
    $\mathbf{e}_p \leftarrow f_{sim}(\mathcal{B}[k_p])$
    $\mathbf{e}_n \leftarrow f_{sim}(\mathcal{B}[k_n])$
    `// Compute the triplet loss`
    $\mathcal{L} \leftarrow \max\left(0, ||\mathbf{e}_a - \mathbf{e}_p||_2^2 - ||\mathbf{e}_a - \mathbf{e}_n||_2^2 + m\right)$
    `// Optimization step`
    optimizer.zero_grad()
    $\mathcal{L}$.backward()
    optimizer.step()
  end
end
return f_{sim}

3.4 Ensemble construction

Our basic algorithm is shown in the diagram 4

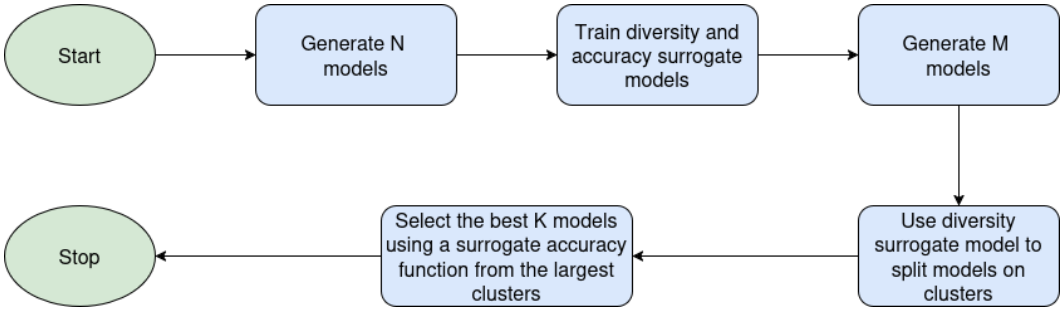

Fig. 4: The main algorithm of our model.

Algorithm 2 Selection of the Best Ensemble Models Using Surrogate Models and DBSCAN

Input: f_{div}: an untrained surrogate diversity model;
f_{acc}: an untrained surrogate accuracy model;
$\mathcal{B}$: a set of architectures of pretrained models (in NAS-Bench-101 format);
N: the number of models to be generated for inference;
K: the required number of models in the ensemble;
ε: the neighborhood radius for DBSCAN;
$minPts$: the minimum number of points required to form a dense region.
Output: $\mathcal{M}^*$: the set of K best models.

// Train surrogate models
Train the model f_{div} using $\mathcal{B}$ Train the model f_{acc} using $\mathcal{B}$

// Generate embeddings for N models
$E \leftarrow \varnothing$ **for** $i \leftarrow 1$ **to** N **do**
  Generate architecture M_i
  Compute embedding $e_i \leftarrow f_{\text{div}}(M_i)$
  $E \leftarrow E \cup \{e_i\}$
end

// Clustering with DBSCAN
$clusters \leftarrow \text{DBSCAN}(E, \varepsilon, minPts)$ Sort the clusters in descending order by their sizes

// Select the best model from each cluster
$\mathcal{M}^* \leftarrow \varnothing$ $K \leftarrow \max(K, |clusters|)$ **for** $i \leftarrow 1$ **to** K **do**
  $acc_i \leftarrow \varnothing$ **foreach** *model* $M_{ij} \in clusters[i]$ **do**
    Compute accuracy score $acc_{ij} \leftarrow f_{\text{acc}}(M_{ij})$
    Add acc_{ij} to the set acc_i
  end
  Compute $j^* = \arg\max_j \{acc_{ij}\}$
  $\mathcal{M}^* \leftarrow \mathcal{M}^* \cup \{M_{ij^*}\}$
end
return $\mathcal{M}^*$

The process begins by training two surrogate models. The diversity model f_{div} learns to map network architectures to a latent space where similar architectures are positioned close together. This is achieved by training on the pretrained architectures $\mathcal{B}$ using their topological properties and predictions on didicated validation dataset by using similarity matrix. Concurrently, the accuracy model f_{acc} is trained to predict validation accuracy from architectural features.

The algorithm then generates N candidate architectures and computes their latent representations using f_{div}. These embeddings are clustered using DBSCAN, which offers three key advantages: 1) automatic determination of cluster count, 2) identification of arbitrarily-shaped clusters, and 3) inherent noise handling. The resulting clusters are sorted by size to prioritize robust architectural patterns.

From each of the top-K clusters (where K is constrained by the actual number of clusters), the model with highest predicted accuracy is selected using f_{acc}. The final ensemble $\mathcal{M}^*$ comprises these optimal representatives, ensuring both diversity (through cluster separation) and accuracy (via intra-cluster selection).

4 Computational Experiment

In this section we present the experimental results as well as the metrics used for comparison. In Section 4.1, we describe the dataset collection procedure. Section 4.2 details the training setup for the surrogate functions. Finally, in Section 4.3, we compare our proposed method against DeepEnsemble and Random Search.

4.1 Dataset Construction

The models in our dataset were generated according to the following algorithm:

1. Split the CIFAR-10 dataset into training and validation subsets with an 80%/20% ratio.
2. Sample "normal" and "reduction" cells from the DARTS search space, ensuring each node has exactly two incoming edges from previous nodes.
3. Assemble a DARTS architecture consisting of two normal cells followed by one reduction cell.
4. Train each sampled architecture on the training subset.
5. For each trained model, record:
   - its architectural description,
   - its predictions on the validation set,
   - its validation accuracy.

Table 1 summarizes the training hyperparameters for all sampled models:

Tab. 1: Training Hyperparameters for Dataset Models

Hyperparameter	Value
Number of epochs	80
Optimizer	Adam
Learning rate	0.001
Training set size	48 000
Validation set size	12 000
Batch size	256

The resulting pool of models exhibited the following distributions:

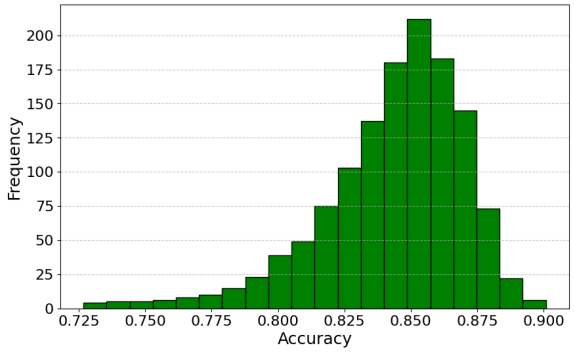

Fig. 5: Distribution of model accuracies

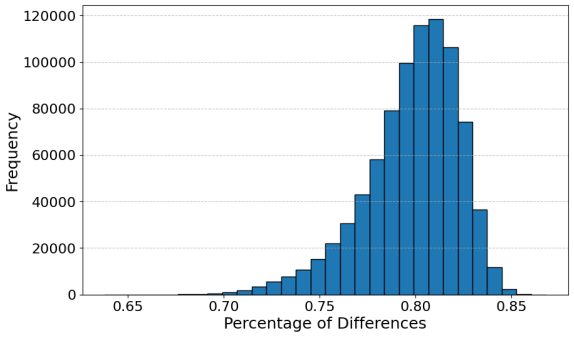

Fig. 6: Distribution of inter-model diversity

4.2 Training of the Surrogate Functions

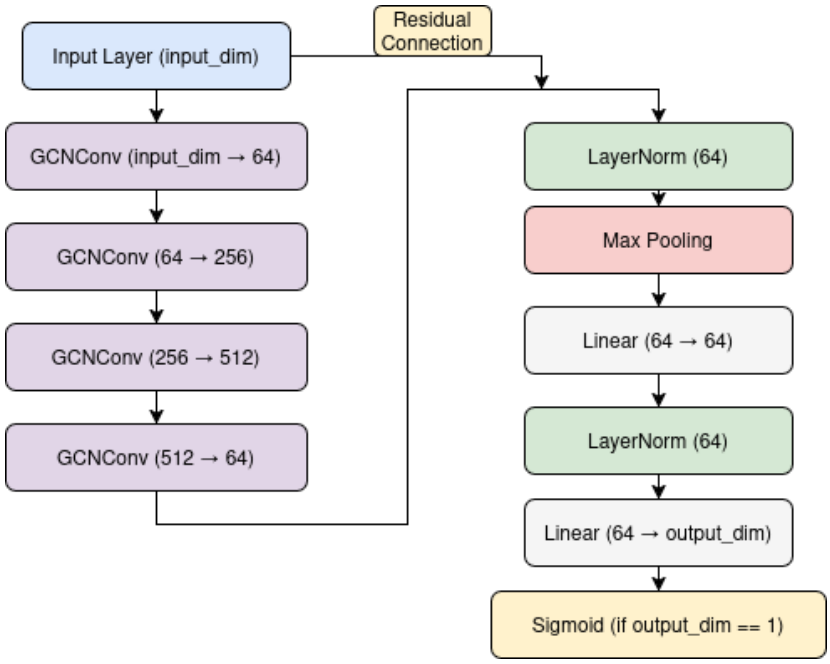

Fig. 7: Architecture of the surrogate function (GCN with 4 convolutional layers and 2 fully connected output layers)

We employed a graph convolutional network (GCN) with four convolutional layers and two fully connected layers at the output (see Figure 7). In both surrogate training regimes, the learning rate was halved every 5 epochs.

For the *surrogate model for architecture diversity*, we set the output dimensionality to 16. Its training hyperparameters were:

Tab. 2: Training Hyperparameters for Diversity Surrogate

Hyperparameter	Value
Number of epochs	**30**
Optimizer	**Adam**
Learning rate	**0.001**
Training set size	**1040**
Validation set size	**260**

For the *surrogate model for architecture accuracy*, we set the output dimensionality to 1 and applied a sigmoid activation to produce a predictive accuracy score. Its training hyperparameters were:

Tab. 3: Training Hyperparameters for Accuracy Surrogate

Hyperparameter	Value
Number of epochs	10
Optimizer	Adam
Learning rate	0.0002
Training set size	1040
Validation set size	260

Fig. 8: Training curve of the surrogate model for diversity

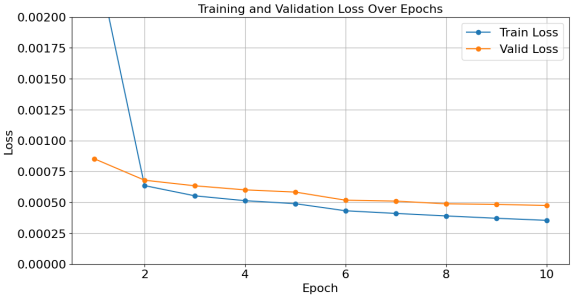

Fig. 9: Training curve of the surrogate model for accuracy

4.3 Experimental Results on Ensemble Construction

References

[1] P.N. Suganthan Ye Ren, Le Zhang. Ensemble classification and regression-recent developments, applications and future directions [review article]. *IEEE Computational Intelligence Magazine*, 11(1):41–53, Feb 2016. ISSN 1556-603X. 10.1109/mci.2015.2471235. URL https://doi.org/10.1109/mci.2015.2471235.

[2] Lars Kai Hansen and Peter Salamon. Neural network ensembles. *IEEE Trans. Pattern Anal. Mach. Intell.*, 12(10): 993–1001, 1990. 10.1109/34.58871.

[3] Sheheryar Zaidi, Arber Zela, Thomas Elsken, Chris C. Holmes, Frank Hutter, and Yee Whye Teh. Neural ensemble search for uncertainty estimation and dataset shift. In Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan, editors, *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pages 7898–7911, 2021. URL https://proceedings.neurips.cc/paper/2021/hash/41a6fd31aa2e75c3c6d427db3d17ea80-Abstract.html.

[4] Esteban Real, Sherry Moore, Andrew Selle, Saurabh Saxena, Yutaka Leon Suematsu, Jie Tan, Quoc V Le, and Alexey Kurakin. Large-scale evolution of image classifiers. In *International conference on machine learning*, pages 2902–2911. PMLR, 2017.

[5] Esteban Real, Alok Aggarwal, Yanping Huang, and Quoc V Le. Regularized evolution for image classifier architecture search. In *Proceedings of the aaai conference on artificial intelligence*, volume 33, pages 4780–4789, 2019.

[6] Barret Zoph and Quoc V. Le. Neural architecture search with reinforcement learning. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017. URL https://openreview.net/forum?id=r1Ue8Hcxg.

[7] Sirui Xie, Hehui Zheng, Chunxiao Liu, and Liang Lin. Snas: stochastic neural architecture search. *arXiv preprint arXiv:1812.09926*, 2018.

[8] Yuqiao Liu, Yanan Sun, Bing Xue, Mengjie Zhang, Gary G. Yen, and Kay Chen Tan. A survey on evolutionary neural architecture search. *IEEE Trans. Neural Networks Learn. Syst.*, 34(2):550–570, 2023. 10.1109/TNNLS.2021.3100554.

[9] Haifeng Jin, Qingquan Song, and Xia Hu. Auto-keras: An efficient neural architecture search system. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 1946–1956, 2019.

[10] Kirthevasan Kandasamy, Willie Neiswanger, Jeff Schneider, Barnabas Poczos, and Eric P Xing. Neural architecture search with bayesian optimisation and optimal transport. *Advances in neural information processing systems*, 31, 2018.

[11] Ankit Kumar Saroj Kumar Pandey Neeraj varshney Teekam Singh Chetan Swarup, Kamred Udham Singh. Brain tumor detection using cnn, alexnet amp;amp; googlenet ensembling learning approaches. *Electronic Research Archive*, 31(5): 2900–2924, 2023. ISSN 2688-1594. 10.3934/era.2023146. URL https://doi.org/10.3934/era.2023146.

[12] Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. *Advances in neural information processing systems*, 30, 2017.

[13] Hanxiao Liu, Karen Simonyan, and Yiming Yang. DARTS: differentiable architecture search. *CoRR*, abs/1806.09055, 2018. 10.48550/arxiv.1806.09055. URL http://arxiv.org/abs/1806.09055.

[14] Yao Shu, Yizhou Chen, Zhongxiang Dai, and Bryan Kian Hsiang Low. Neural ensemble search via bayesian sampling. In James Cussens and Kun Zhang, editors, *Proceedings of the Thirty-Eighth Conference on Uncertainty in Artificial Intelligence*, volume 180 of *Proceedings of Machine Learning Research*, pages 1803–1812. PMLR, 01–05 Aug 2022. URL https://proceedings.mlr.press/v180/shu22a.html.

[15] Haibin Ling Minghao Chen, Jianlong Fu. One-shot neural ensemble architecture search by diversity-guided search space shrinking. *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 16525–16534, Jun 2021. 10.1109/cvpr46437.2021.01626. URL https://doi.org/10.1109/cvpr46437.2021.01626.

[16] Zhichao Lu, Ran Cheng, Shihua Huang, Haoming Zhang, Changxiao Qiu, and Fan Yang. Surrogate-assisted multi-objective neural architecture search for real-time semantic segmentation. *CoRR*, abs/2208.06820, 2022. 10.48550/ARXIV.2208.06820.

[17] Zhichao Lu, Kalyanmoy Deb, Erik D. Goodman, Wolfgang Banzhaf, and Vishnu Naresh Boddeti. Nsganetv2: Evolutionary multi-objective surrogate-assisted neural architecture search. In Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm, editors, *Computer Vision - ECCV 2020 - 16th European Conference, Glasgow, UK, August 23-28, 2020, Proceedings, Part I*, volume 12346 of *Lecture Notes in Computer Science*, pages 35–51. Springer, 2020. 10.1007/978-3-030-58452-8_3.

[18] Maria G. Baldeon Calisto and Susana K. Lai-Yuen. Emonas-net: Efficient multiobjective neural architecture search using surrogate-assisted evolutionary algorithm for 3d medical image segmentation. *Artif. Intell. Medicine*, 119:102154, 2021. 10.1016/J.ARTMED.2021.102154.

[19] Ferrante Neri Yu Xue, Zhenman Zhang. Similarity surrogate-assisted evolutionary neural architecture search with dual encoding strategy. *Electronic Research Archive*, 32(2):1017–1043, 2024. ISSN 2688-1594. 10.3934/era.2024050. URL https://doi.org/10.3934/era.2024050.

[20] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017. URL https://openreview.net/forum?id=SJU4ayYgl.

[21] Wei Wen, Hanxiao Liu, Yiran Chen, Hai Li, Gabriel Bender, and Pieter-Jan Kindermans. Neural predictor for neural architecture search. In *European conference on computer vision*, pages 660–676. Springer, 2020.

[22] Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 815–823, 2015.

[23] Xuanyi Dong and Yi Yang. Nas-bench-201: Extending the scope of reproducible neural architecture search. *arXiv preprint arXiv:2001.00326*, 2020.

[24] Chris Ying, Aaron Klein, Esteban Real, Eric Christiansen, Kevin Murphy, and Frank Hutter. Nas-bench-101: Towards reproducible neural architecture search. *CoRR*, abs/1902.09635, 2019. 10.48550/arxiv.1902.09635. URL http://arxiv.org/abs/1902.09635.

[25] Wei Wen, Hanxiao Liu, Hai Li, Yiran Chen, Gabriel Bender, and Pieter-Jan Kindermans. Neural predictor for neural architecture search. *CoRR*, abs/1912.00848, 2019. 10.48550/arxiv.1912.00848. URL http://arxiv.org/abs/1912.00848.