**Research Article**

Alexandr Udeneev, Petr Babkin, and Oleg Bakhteev

# Surrogate assisted diversity estimation in neural ensemble search

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

**Abstract:** Since most neural architecture search (NAS) methods are computationally intensive, extending them to neural ensemble search (NES) – which involves jointly optimizing both individual architectures and their ensemble configurations – can lead to a combinatorial increase in the search space, often resulting in prohibitively high computational costs. To address this, we introduce a dual-surrogate formulation: candidate architectures are represented as graphs, and two surrogate models are trained separately to predict accuracy and ensemble diversity. Their combined estimates guide an NES framework that efficiently identifies architectures that are both individually strong and collectively diverse. Our final ensemble achieves near-state-of-the-art accuracy on FashionMNIST, CIFAR-10, and CIFAR-100, surpassing standard baselines such as Deep Ensembles and random architecture search, and matches the performance of top-performing methods in the literature.

**Keywords:** neural ensemble search, ensemble diversity, surrogate function, triplet loss.

## 1 Introduction

Neural network ensembles often demonstrate better accuracy, improved robustness, and more reliable uncertainty estimation 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].

One of the pioneer works for NES is DeepEns [12], which trains the same neural network architecture multiple times with different random initializations and combines their predictions. While this method is simple to implement and tune, its effectiveness is limited because there is no guarantee that the resulting ensemble will be sufficiently diverse, which can reduce potential gains in accuracy and uncertainty estimation. More sophisticated adaptation of NAS techniques are presented in some recent works [3, 13, 14], which are designed to efficiently combine multiple networks into an ensemble. However, these methods lack explicit, structured control over ensemble diversity: diversity either emerges implicitly through optimization objectives or is guided by heuristic similarity measures, without a dedicated mechanism to balance it against individual model accuracy.

Our research also adapts ideas from NAS for NES, specifically utilizing a surrogate function [15–17]. Some modern NAS methods widely use surrogate functions to estimate architecture quality without requiring full model training. These functions significantly reduce computational costs, expanding the applicability of such methods. For example, in [15], evolutionary algorithms were proposed in combination with surrogate models for real-time semantic segmentation. In [17], a Surrogate-assisted Multiobjective Evolutionary-based Algorithm (SaMEA) is used for 3D medical image segmentation.

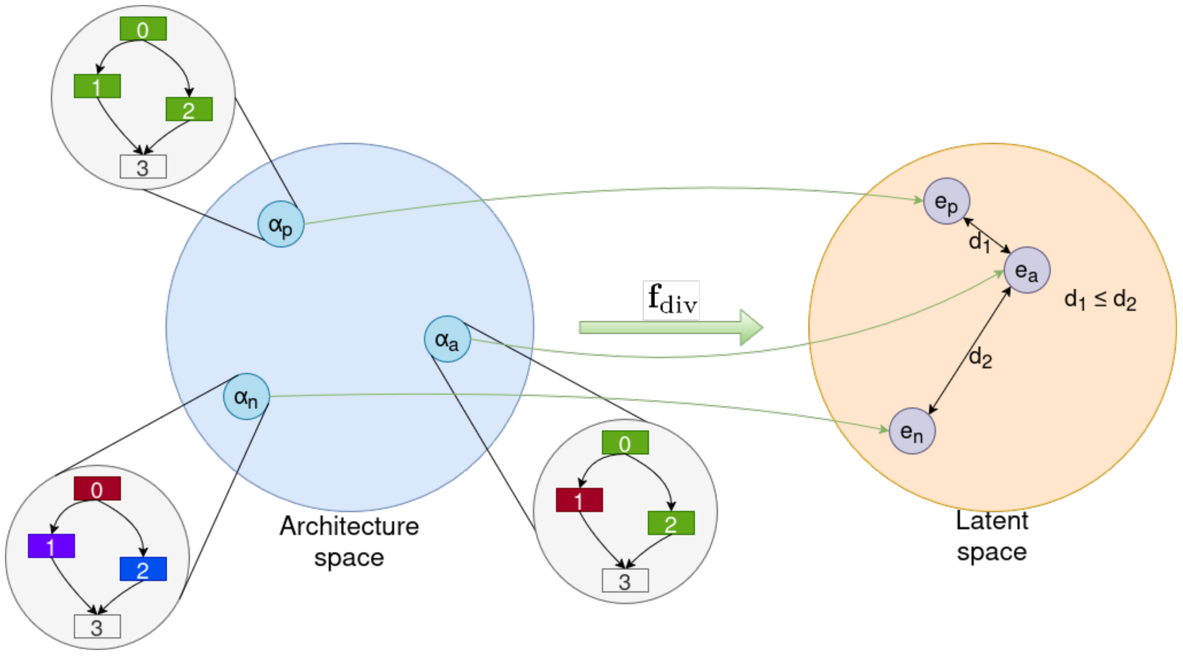

**Fig. 1:** Visualization of the surrogate diversity function. Architecture space showing different DARTS-like architectures represented as graphs. A surrogate diversity function transforms this space into latent space. In latent space where similar architectures (e.g. $\mathbf{e}_a$ and $\mathbf{e}_p$) are mapped close together, while dissimilar architectures (e.g., $\mathbf{e}_a$ and $\mathbf{e}_n$) are mapped farther apart, satisfying $d_1 \leq d_2$. Geometric organization in a latent space makes it possible to effectively evaluate diversity without the need for pre-training models with these architectures.

To address the challenges of NES, we propose a dual-surrogate framework that decouples the search into two predictive tasks: estimating individual architecture accuracy and predicting each architecture's contribution to ensemble diversity. Inspired by surrogate-assisted NAS [15, 17], we model candidate architectures as graphs and train lightweight surrogate models to capture these properties. Crucially, diversity is represented a priori via geometric relationships in a latent space [18]. As illustrated in Fig. 1, the surrogate diversity function maps architectures from the discrete architecture space into a continuous latent space, where similar architectures are embedded close to each other, while dissimilar ones are separated by larger distances. This geometric organization enables efficient and explicit estimation of ensemble diversity without requiring costly pre-training of candidate architectures. For graph-based surrogate modeling, Graph Attention Networks (GATs) [19, 20] are used, with Triplet Loss [21] employed to structure the latent space according to diversity. We evaluate our approach on FashionMNIST, CIFAR-10, and CIFAR-100. The constructed ensembles consistently outperform standard baselines, including DeepEns and random architecture search, while matching the performance of recent state-of-the-art NES methods. These results highlight that explicitly modeling architectural diversity, combined with surrogate-guided search, enables the efficient construction of high-performance ensembles without incurring prohibitive computational costs.

Main Contributions:

1. We propose a way for training the surrogate function to predict the diversity of architectures.
2. We present the first surrogate-assisted framework for neural ensemble construction that jointly optimizes predictive performance and architectural diversity.
3. We rigorously validate that our method matches or exceeds the performance of both standard ensemble construction baselines (Deep Ensembles, random architecture search) and state-of-the-art neural ensemble search approaches across FashionMNIST, CIFAR-10, and CIFAR-100 benchmarks, while operating at a fraction of their computational cost.

## 2 Problem statement

### 2.1 Neural Architecture Search

We define:

$$\mathcal{V} = \{\mathbf{x}_1, \dots, \mathbf{x}_N\} \subset \mathbb{R}^d, \quad \mathcal{O} = \{o_1, \dots, o_K\}, \;\; o_k : \mathbb{R}^d \to \mathbb{R}^d, \quad \mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$$

where $\mathcal{V}$ is the set of node feature-vectors in $\mathbb{R}^d$, $\mathcal{O}$ is a finite set of elementwise operations (e.g., convolutions, poolings), and $\mathcal{E}$ is the set of directed edges between nodes.

We then define the space of feasible architectures as the set of all acyclic directed graphs over $\mathcal{V}$ with operations on edges:

$$\mathcal{A} = \Big\{ (\mathcal{V}, E) \;\Big|\; E \subseteq \mathcal{E},\; (V, E) \text{ is a DAG},\; \forall (u \to v) \in E,\; \text{assign } o_{u\to v} \in \mathcal{O} \Big\}.$$

Equivalently, an architecture $\alpha \in \mathcal{A}$ can be represented by

$$\big(V, \{(u \to v,\, o_{u\to v}) \mid (u \to v) \in E\}\big),$$

where $E$ induces an acyclic graph on $\mathcal{V}$ and each edge $(u \to v)$ is labeled by an operation $o_{u\to v} \in \mathcal{O}$.

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 $\boldsymbol{\alpha}^*$ that minimizes $\mathcal{L}_{val}(\boldsymbol{\alpha}^*, \boldsymbol{\omega}^*)$, under the constraint that the weights are obtained by minimizing the training loss:

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

This can be expressed as the following optimization problem:

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

The primary challenge in this optimization lies in the immense search space of possible architectures (e.g., in DARTS [22], 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 $\boldsymbol{\alpha} \in \mathcal{A}$ as a network architecture and $\omega(\alpha)$ as its corresponding parameters. The action of this network on an input $x$ is denoted by $f_{\boldsymbol{\alpha}}(\boldsymbol{x}, \boldsymbol{\omega}(\boldsymbol{\alpha}))$. Let $S \subset \mathcal{A}$ be a subset of architectures. Then, the NES problem can be formally described as follows:

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

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

## 3 Method

In this work, we consider the transformation of the architecture space proposed in DARTS [22] for application in Graph Attention Networks (GAT) (see Section 3.1). In Section 3.2, we present the architecture of the surrogate functions and describe its working principle, while Section 3.3 provides a detailed discussion of the ensemble construction method based on this surrogate functions, explaining how surrogate predictions are used to ensure architectural diversity within the ensemble.

The proposed approach enables ensembles that optimally trade off between predictive accuracy and architectural diversity.

## 3.1 Architecture Search Space

A critical aspect of our methodology involves constructing a training dataset for the surrogate model. While we adopt the DARTS framework [22], which defines normal and reduction cells, our approach diverges fundamentally in how these cells are generated. Specifically, instead of optimizing cells through continuous relaxation, we generate both cell types via discrete random sampling.

Each cell is represented as a directed acyclic graph (DAG) comprising $n$ nodes and $m$ edges, where every edge corresponds to an operation selected from the DARTS operation set $\mathcal{O}$. Formally, a cell architecture is defined as:

$$\alpha_{\text{cell}} = (G, \mathbf{o}), \quad G = (\mathcal{V}, \mathcal{E}), \quad \mathbf{o} = \{o_e\}_{e \in \mathcal{E}}, \quad o_e \in \mathcal{O}, \tag{3}$$

where $\mathcal{V}$ denotes the set of $n$ latent nodes, $\mathcal{E}$ represents the $m$ directed edges, and $\mathcal{O}$ is the predefined operation space.

The full architecture space $\mathcal{A}$ consists of pairs of such DAGs (denoted as normal and reduction cells):

$$\mathcal{A} = \{(G_1, \mathbf{o_1}) \times (G_2, \mathbf{o_2}) \mid G_1, G_2 \text{ are DAGs}, |\mathcal{V}_1| = |\mathcal{V}_2| = n, |\mathcal{E}| = m, \mathbf{o}_1, \mathbf{o}_2 \in \mathcal{O}^m\}. \tag{4}$$

To train the surrogate model, we generate $N$ architectures, evaluate their performance on a fixed validation dataset, and compile the training dataset:

$$\mathcal{D}_{\text{train}} = \{(\alpha_i, \mathbf{y}_i, \text{acc}_i)\}_{i=1}^{N}, \tag{5}$$

where $\mathbf{y}_i$ is the vector of predictions from architecture $\alpha_i$ on the validation set, and $\text{acc}_i$ is its validation accuracy.

Unlike DARTS, which employs continuous relaxation and gradient-based optimization, our method relies on discrete sampling to explore the search space. This strategy enables broader diversity in architecture generation at the expense of computational efficiency.

Let $n = 5, m = 10$ and use DARTS search space with 7 operations, then the total number of unique cell architectures is computed as:

$$|\mathcal{A}_{\text{cell}}| = \binom{2}{2}\binom{3}{2}\binom{4}{2}\binom{5}{2} \cdot |\mathcal{O}|^8, \quad |\mathcal{O}| = 7, \tag{6}$$

resulting in a complete architecture space of:

$$|\mathcal{A}| = |\mathcal{A}_{\text{cell}}|^2 \approx 10^{18}. \tag{7}$$

This vast combinatorial space renders exhaustive search computationally intractable, necessitating efficient surrogate-guided exploration.

## 3.2 Surrogate Function

In order to construct the ensemble described in Section 3.3, we need to predict both the performance and the diversity of candidate architectures. Due to the enormous size of the architecture space, obtaining these characteristics via full training is infeasible. To address this, we employ surrogate models: given an architectural representation, they predict key properties of neural networks. Formally, we define

$$\mathbf{f} : \mathcal{A} \to \mathbb{R}^d,$$

where $d$ is the dimension of the latent space. In particular, we consider two surrogates:

$$f^{\theta}_{\text{acc}} : \mathcal{A} \to \mathbb{R}, \quad \mathbf{f}^{\theta}_{\text{div}} : \mathcal{A} \to \mathbb{R}^d.$$

Each architecture is represented as a directed acyclic graph (DAG), where nodes correspond to latent representations and edges to operations, similar to NAS-Bench-201 [23]. For surrogate modeling, we adopt an alternative graph representation, inspired by NAS-Bench-101 [24]: nodes represent operations and edges their corresponding latent embeddings. Conversion is done as follows:

1. Each edge of the original graph is transformed into a node in the new graph, labeled by the operation present on that edge.
2. An oriented edge is drawn from operation $o_i$ to $o_j$ if in the original graph $o_i$ acts from node $x$ to $y$ and $o_j$ acts from $y$ to $z$.

Operations are encoded as one-hot vectors. The surrogates themselves are Graph Attention Networks (GATs) with $N$ sequential layers, residual connections, GraphNorm, a global pooling aggregation, and lightweight output heads (Figure 2).

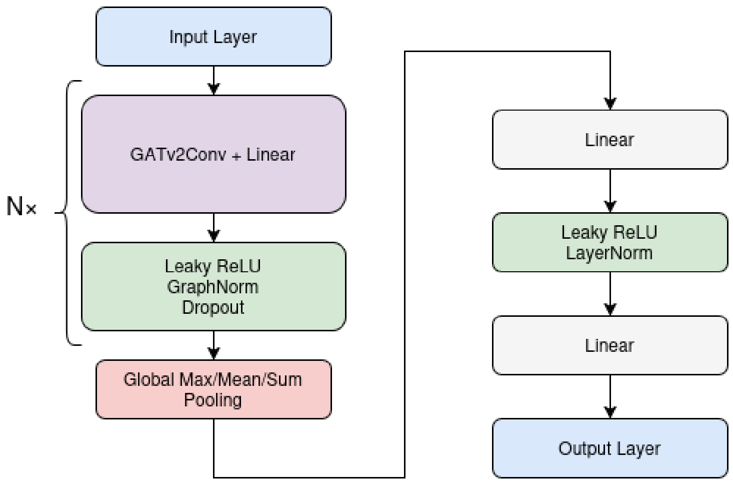

**Fig. 2:** Surrogate architecture: $N$ GAT layers with residual connections and GraphNorm, global pooling, and output heads for accuracy and similarity embeddings.

The accuracy surrogate $f^{\theta}_{\text{acc}}$ is trained via supervised regression using the mean squared error:

$$\theta^* = \arg\min_{\theta} \sum_i \left(f^{\theta}_{\text{acc}}(\mathbf{a}_i) - y_i\right)^2. \tag{8}$$

In contrast, the diversity surrogate $\mathbf{f}^{\theta}_{\text{div}}$ cannot be trained directly with supervised learning, since no ground-truth latent embeddings exist. Instead, we optimize it using a triplet loss on architecture embeddings.

Let us consider a set of $N$ trained models $\{M_1, \ldots, M_N\}$ and a fixed validation dataset of $K$ examples. Denote the predictions of model $M_i$ on the validation dataset by the vector

$$\mathbf{y}^{(i)} = \left(y^{(i)}_1, \ldots, y^{(i)}_K\right).$$

We construct a diversity matrix $\mathbf{D} \in [0,1]^{N \times N}$, where each entry

$$d_{ij} = \frac{1}{K} \sum_{k=1}^{K} \mathbb{I}\left(y^{(i)}_k = y^{(j)}_k\right)$$

measures the fraction of matching predictions between models $M_i$ and $M_j$.

For the purposes of training with triplet loss, $\mathbf{D}$ is discretized into a matrix $\mathbf{M} \in \{-1, 0, 1\}^{N\times N}$ according to upper and lower quantile thresholds $q_p$ and $q_n$:

$$\mathbf{M}_{ij} = \begin{cases} 1, & \text{if } d_{ij} > q_p, \\ -1, & \text{if } d_{ij} < q_n, \\ 0, & \text{otherwise.} \end{cases}$$

**Algorithm 1:** Training the surrogate model for architecture diversity

```
Input: $\mathbf{f}_{div}$: an untrained surrogate model; $\mathcal{B}$: a set of architectures of pretrained models; $N$: 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 $\mathbf{f}_{div}$
for $i \leftarrow 1$ to $n$ do
    for $j \leftarrow 1$ to $N$ do
        $\mathcal{P}_j \leftarrow \{k \mid \mathbf{M}[j,k] = 1\}$            // Sample a positive example
        $k_p \leftarrow \text{UniformSample}(\mathcal{P}_j)$                  // Pick a positive index
        $\mathcal{N}_j \leftarrow \{k \mid \mathbf{M}[j,k] = -1\}$           // Sample a negative example
        $k_n \leftarrow \text{UniformSample}(\mathcal{N}_j)$                  // Pick a negative index
                                                                              // Compute embeddings
        $\mathbf{e}_a \leftarrow \mathbf{f}_{div}(\mathcal{B}[j])$                             // Anchor embedding
        $\mathbf{e}_p \leftarrow \mathbf{f}_{div}(\mathcal{B}[k_p])$                           // Positive embedding
        $\mathbf{e}_n \leftarrow \mathbf{f}_{div}(\mathcal{B}[k_n])$                           // Negative embedding

        $\mathcal{L} \leftarrow \max\left(0, ||\mathbf{e}_a - \mathbf{e}_p||_2^2 - ||\mathbf{e}_a - \mathbf{e}_n||_2^2 + m\right)$   // Compute the triplet loss

                                                                              // Optimization step
        optimizer.zero_grad()
        $\mathcal{L}$.backward()
        optimizer.step()
    end
end
return $\mathbf{f}_{div}$
```

The training procedure of $\mathbf{f}_{\text{div}}$ (Algorithm 1) proceeds as follows. For each epoch and for each anchor architecture $\alpha_a$, we first sample one positive $\alpha_p$ and one negative $\alpha_n$ example uniformly from the corresponding rows of the discretized similarity matrix $\mathbf{M}$. Next, the anchor, positive, and negative architectures are embedded via the surrogate function $\mathbf{f}^{\theta}_{\text{div}}$:

$$\mathbf{e}_a = \mathbf{f}^{\theta}_{\text{div}}(\alpha_a), \quad \mathbf{e}_p = \mathbf{f}^{\theta}_{\text{div}}(\alpha_p), \quad \mathbf{e}_n = \mathbf{f}^{\theta}_{\text{div}}(\alpha_n).$$

We then compute the triplet loss with margin $m$:

$$\mathcal{L} = \max\left(\|\mathbf{e}_a - \mathbf{e}_p\|_2^2 - \|\mathbf{e}_a - \mathbf{e}_n\|_2^2 + m, 0\right),$$

and perform the standard optimization steps: zeroing the gradients, backpropagating $\mathcal{L}$, and updating the model parameters with the chosen optimizer.

This optimization scheme encourages the surrogate to map similar architectures closer together in the latent space, while dissimilar architectures are separated by at least margin $m$. Diversity between models can also be quantified using other metrics in the output space, such as fraction of identical predictions or distances/divergences between output distributions (e.g., Jensen-Shannon divergence or Hellinger distance), yielding an $N \times N$ diversity matrix.

## 3.3 Ensemble construction

The algorithm 2 begins by initializing three sets: $\mathcal{A}_{pool}$, which stores candidate architectures whose predicted accuracy exceeds a predefined threshold $\alpha$; $\mathcal{E}_{pool}$, which holds the corresponding diversity embeddings; and $\mathcal{S}_{pool}$, which contains the predicted accuracy scores obtained from surrogate models.

After this, $N$ random candidate architectures, denoted by $\mathcal{A}_{cand}$, is generated. For each architecture, both the predicted accuracy and a representation in the latent diversity space are computed using the surrogate functions $f_{acc}$ and $\mathbf{f}_{div}$, respectively. Only those architectures with predicted accuracy above the threshold $\alpha$ are added to the pool.

**Algorithm 2:** Greedy Ensemble Construction

```
Input: $K$: ensemble size; $M$: minimum pool size ($M \geq K$); $N$: number of architectures generated
       per iteration; $\alpha$: accuracy threshold; $f_{\text{acc}}$: surrogate model for accuracy prediction; $f_{\text{div}}$:
       surrogate model for diversity embeddings.
Output: $\mathcal{A}_{\text{best}}$: set of $K$ selected architectures.
1  $\mathcal{A}_{\text{pool}} \leftarrow \emptyset$                                // Initialize architecture pool
2  $\mathcal{E}_{\text{pool}} \leftarrow \emptyset$                                // Initialize embedding pool
3  $\mathcal{S}_{\text{pool}} \leftarrow \emptyset$                                // Initialize accuracy score pool
4  $\mathcal{A}_{\text{cand}} \leftarrow \text{GenerateArchs}(N)$          // Generate N random candidate architectures
5  $\mathbf{s} \leftarrow f_{\text{acc}}(\mathcal{A}_{\text{cand}})$                          // Predict accuracy scores
6  $\mathbf{e} \leftarrow f_{\text{div}}(\mathcal{A}_{\text{cand}})$                          // Predict diversity embeddings
7  for $(a, s, e) \in zip(\mathcal{A}_{cand}, \mathbf{s}, \mathbf{e})$ do
8      if $s \geq \alpha$ then
9          $\mathcal{A}_{\text{pool}} \leftarrow \mathcal{A}_{\text{pool}} \cup \{a\}$
10         $\mathcal{E}_{\text{pool}} \leftarrow \mathcal{E}_{\text{pool}} \cup \{e\}$
11         $\mathcal{S}_{\text{pool}} \leftarrow \mathcal{S}_{\text{pool}} \cup \{s\}$
12     end
13 end
14 $\mathcal{A}_{\text{best}} \leftarrow \emptyset$                                // Initialize selected set
15 $\mathcal{E}_{\text{selected}} \leftarrow \emptyset$                            // Initialize selected embeddings
16 $i^* \leftarrow \arg\max_i \mathcal{S}_{\text{pool}}[i]$                     // Index of highest accuracy
17 $\mathcal{A}_{\text{best}} \leftarrow \mathcal{A}_{\text{best}} \cup \{\mathcal{A}_{\text{pool}}[i^*]\}$
18 $\mathcal{E}_{\text{selected}} \leftarrow \mathcal{E}_{\text{selected}} \cup \{\mathcal{E}_{\text{pool}}[i^*]\}$
19 Remove index $i^*$ from $\mathcal{A}_{\text{pool}}, \mathcal{E}_{\text{pool}}, \mathcal{S}_{\text{pool}}$          // Remove selected from pools
20 while $|\mathcal{A}_{best}| < K$ and $|\mathcal{A}_{pool}| > 0$ do
21     foreach $i \in \{0, \ldots, |\mathcal{E}_{pool}| - 1\}$ do
22         $d_i \leftarrow \frac{1}{|\mathcal{E}_{\text{selected}}|} \sum_{e_{\text{sel}} \in \mathcal{E}_{\text{selected}}} \|\mathcal{E}_{\text{pool}}[i] - e_{\text{sel}}\|_2$     // Mean distance to selected set
23     end
24     $i^* \leftarrow \arg\max_i d_i$                                  // Most diverse architecture
25     $\mathcal{A}_{\text{best}} \leftarrow \mathcal{A}_{\text{best}} \cup \{\mathcal{A}_{\text{pool}}[i^*]\}$
26     $\mathcal{E}_{\text{selected}} \leftarrow \mathcal{E}_{\text{selected}} \cup \{\mathcal{E}_{\text{pool}}[i^*]\}$
27     Remove index $i^*$ from $\mathcal{A}_{\text{pool}}, \mathcal{E}_{\text{pool}}, \mathcal{S}_{\text{pool}}$      // Remove selected from pools
28 end
29 return $\mathcal{A}_{best}$
```

Once the pool construction step is complete, the ensemble is formed using a greedy forward-selection strategy. First, the architecture with the highest predicted accuracy is selected and added to the ensemble, together with its corresponding diversity embedding. The remaining $K-1$ models are then chosen iteratively: for every architecture still in the pool, the mean Euclidean distance in the diversity embedding space to the architectures already included in the ensemble is computed. At each iteration, the architecture with the largest mean distance, i.e., the one most dissimilar from the current ensemble is selected and removed from the pool. This process continues until $K$ architectures have been collected or the pool is exhausted.

## 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 [12] and Random Search [25].

### 4.1 Dataset Construction

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

1. Sample architecture from the DARTS search space, ensuring each node has exactly two incoming edges from previous nodes.
2. Split the dataset into training and validation subsets with an 20%/80% ratio.
3. Train each sampled architecture on the training subset.
4. For each trained model, record:
   - its architectural description,
   - its predictions on the validation set,
   - its validation accuracy.

For training the models, we adopt the hyperparameter configuration from [22], varying only the number of epochs, the number of cells, and the channel width.

Table 1 summarizes the training hyperparameters and results for all sampled models:

**Tab. 1:** Training Hyperparameters and Performance per Dataset

| Dataset | Number of Cells | Initial Width | Num. Epochs | Avg. Accuracy (%) | Avg. Diversity |
|---|---|---|---|---|---|
| **FashionMNIST** | **3** | **16** | **125** | – | – |
| **CIFAR-10** | **8** | **16** | **200** | – | – |
| **CIFAR-100** | **8** | **16** | **200** | – | – |

### 4.2 Training of the Surrogate Functions

In our experiment, each cell contains $n = 5$ nodes, with two outgoing edges per node assigned random operations from $\mathcal{O}$, yielding $m = 10$ edges per cell. The operation set $\mathcal{O}$ includes:

- Separable convolutions (3×3, 5×5)
- Dilated separable convolutions (3×3, 5×5)
- 3×3 max pooling
- 3×3 average pooling
- Identity operation

We employed a graph attention network (GAT) with four convolutional layers and two fully connected layers at the output, i.e. $N = 4$ (see Figure 2). The training of the surrogate functions used a cosine annealing learning rate scheduler.

Full models are constructed by stacking normal and reduction cells in a 2:1 ratio, starting from a predefined number of initial channels. Each model combines one normal and one reduction cell sampled independently from $\mathcal{A}$.

## 4.3 FashionMNIST

For constructing the diversity matrix, we used $q_p = 0.9$ and $q_n = 0.1$. In the diversity regime, the GAT surrogate has an output dimensionality of 16, employs 16 attention heads without dropout, and is trained for 50 epochs with a cosine-annealed learning rate from $5 \times 10^{-4}$ to $1 \times 10^{-5}$. For accuracy prediction, the surrogate outputs a single scalar, uses 16 attention heads, and is trained for 100 epochs with a cosine-annealed learning rate from $5 \times 10^{-3}$ to $1 \times 10^{-4}$. Both models are optimized with Adam on a 2400/600 train/validation split.

Subsequently, the models obtained via the surrogate functions were trained for 125 epochs using stochastic gradient descent (SGD) with a cosine-annealed learning rate, starting from 0.025 and decreasing to 0.001. Training was performed with a weight decay of $3 \times 10^{-4}$, an auxiliary loss weight of 0.4, a network composed of 3 cells with width 16, and a batch size of 96.

Results on FashionMNIST are summarized in Table 2 and Figure 3. The surrogate ensemble achieves slightly lower top-1 accuracy than DeepEns (95.3% vs 95.4%), because its diversity cannot fully manifest on a relatively uniform dataset like FashionMNIST. However, it shows better robustness under adversarial attacks, as the ensemble diversity helps reduce sensitivity to FGSM, BIM, and PGD perturbations.

Table 2 compares standard accuracy metrics, calibration measures (NLL, Oracle NLL, Brier Score, ECE), and ensemble diversity metrics (Ambiguity, Normalized Disagreement, Predictive Disagreement).

Figure 3 illustrates how the surrogate ensemble accuracy degrades under increasing $\varepsilon$ for FGSM, BIM, and PGD attacks. The plots show that, despite a small loss in raw accuracy, the surrogate ensemble consistently outperforms DeepEns under adversarial perturbations, particularly at higher $\varepsilon$ values, demonstrating its improved robustness and adversarial resilience.

**Tab. 2:** FashionMNIST ensemble results.

| Metric | Surrogate Ensemble | DeepEns |
|---|---|---|
| Top-1 Accuracy | $95.3 \pm 0.1\%$ | **95.4%** |
| Average Model Accuracy | $94.7 \pm 0.1\%$ | **95.07** |
| NLL | $0.263 \pm 0.003$ | **0.256** |
| Oracle NLL | $0.199 \pm 0.010$ | 0.207 |
| Brier Score | $0.089 \pm 0.001$ | – |
| ECE | $0.124 \pm 0.002$ | 0.120 |
| Ambiguity | $0.0058 \pm 0.0009$ | – |
| Normalized Disagreement | $0.065 \pm 0.008$ | **1.95** |
| Predictive Disagreement | $0.130 \pm 0.016$ | – |
| Number of models | 3 | 3 |
| Total samples | 10000 | – |

## 4.4 CIFAR10

For constructing the diversity matrix, we used an upper margin of 0.9, a lower margin of 0.1, and measured diversity using the overlap metric.

In the diversity regime, the GAT surrogate has an output dimensionality of 16, employs 16 attention heads without dropout, and is trained for 20 epochs with a cosine-annealed learning rate from $1 \times 10^{-4}$ to

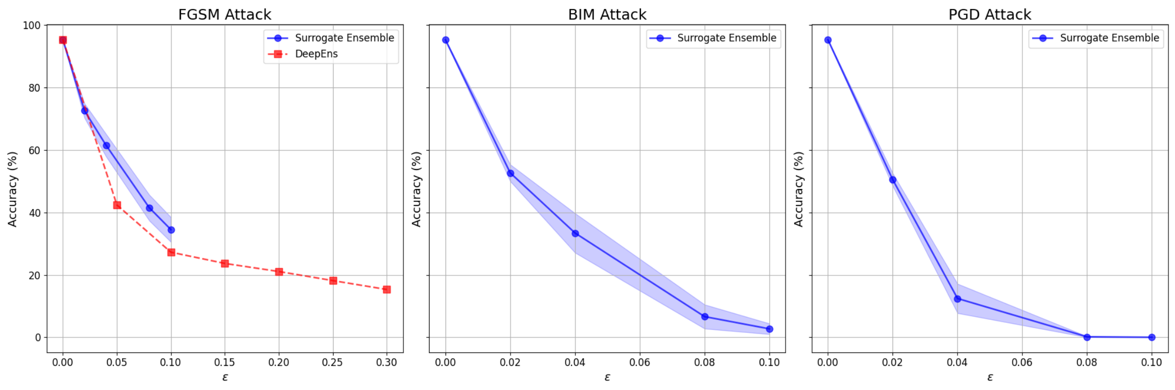

**Fig. 3:** Accuracy of the surrogate ensemble under FGSM, BIM, and PGD attacks across increasing $\varepsilon$. The ensemble maintains higher robustness under stronger perturbations, demonstrating improved adversarial resilience compared to individual models.

$1 \times 10^{-5}$. For accuracy prediction, the surrogate outputs a single scalar, uses 16 attention heads without dropout, and is trained for 100 epochs with a cosine-annealed learning rate from $1 \times 10^{-3}$ to $1 \times 10^{-4}$. Both models are optimized with Adam on a 2400/600 train/validation split.

The models obtained via the surrogate functions were subsequently trained using the same settings as for FashionMNIST 4.3, with minor adjustments: the network consisted of 20 cells of width 36, and training was performed for 600 epochs, while all other hyperparameters, including learning rate schedule, weight decay, auxiliary loss weight, and batch size, remained unchanged.

Table 3 summarizes the evaluation results for the surrogate ensemble and DeepEns on CIFAR10. The surrogate-based ensemble achieves slightly higher top-1 accuracy (97.80% vs. 97.61%) and shows improved diversity metrics, indicating that ensembles constructed via surrogate functions capture a broader range of predictive patterns present in the CIFAR10 dataset.

In terms of adversarial robustness, both methods perform similarly under FGSM and PGD attacks, while the surrogate ensemble shows a small drop under BIM, as illustrated in Figure 4.

**Tab. 3:** CIFAR10 ensemble results.

| Metric | Surrogate Ensemble | DeepEns |
|---|---|---|
| Top-1 Accuracy | $\mathbf{97.80 \pm 0.08\%}$ | $97.61 \pm 0.00\%$ |
| Average Model Accuracy | $96.73 \pm 0.10\%$ | $96.73 \pm 0.00\%$ |
| NLL | $\mathbf{0.208 \pm 0.003}$ | $0.209 \pm 0.000$ |
| Oracle NLL | $\mathbf{0.156 \pm 0.002}$ | $0.158 \pm 0.000$ |
| Brier Score | $\mathbf{0.055 \pm 0.001}$ | $0.056 \pm 0.000$ |
| ECE | $0.136 \pm 0.002$ | $\mathbf{0.134 \pm 0.000}$ |
| Ambiguity | $\mathbf{0.011 \pm 0.001}$ | $0.009 \pm 0.000$ |
| Normalized Disagreement | $\mathbf{0.044 \pm 0.001}$ | $0.042 \pm 0.000$ |
| Predictive Disagreement | $\mathbf{0.087 \pm 0.003}$ | $0.085 \pm 0.000$ |
| Number of models | 5 | 5 |
| Total samples | 10000 | 10000 |

## 4.5 CIFAR100

We used the same configuration as in Section 4.4. Table 4 summarizes the ensemble comparison on CIFAR100. The surrogate-based ensemble achieves higher Top-1 accuracy, lower negative log-likelihood, lower oracle

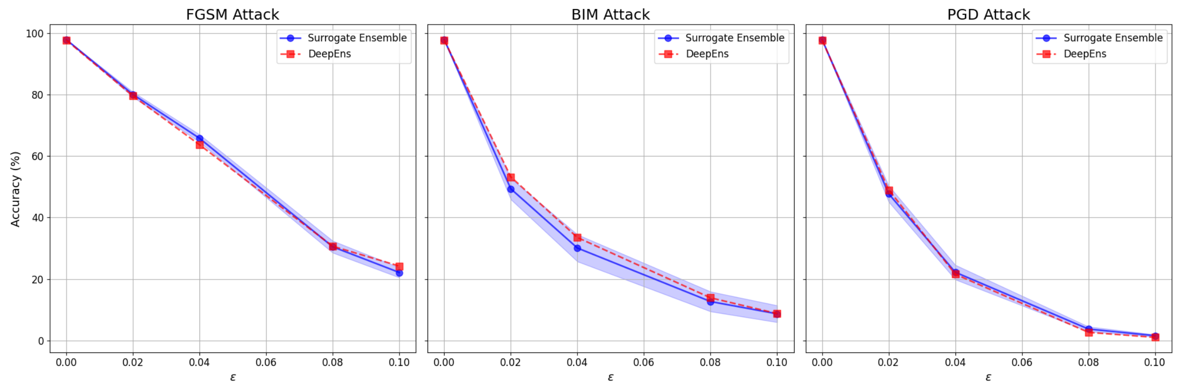

**Fig. 4:** Accuracy of the surrogate ensemble and DeepEns under FGSM, BIM, and PGD attacks across increasing $\varepsilon$ on CIFAR10. The surrogate ensemble consistently outperforms DeepEns, particularly under stronger adversarial perturbations.

NLL, and a better Brier score. These results indicate that models selected via the surrogate functions form a stronger and more diverse ensemble, capturing a broader range of predictive behaviors within the CIFAR100 distribution. DeepEns, however, attains slightly better ECE and disagreement-based metrics, reflecting marginally improved calibration.

Figure 5 shows the ensemble accuracy under adversarial attacks (FGSM, BIM, PGD). Across all perturbation levels, the surrogate-based ensemble maintains slightly higher robustness compared to DeepEns, further supporting the claim that its greater internal diversity leads to better generalization under distribution shifts.

**Tab. 4:** CIFAR100 ensemble results.

| **Metric** | **Surrogate Ensemble** | **DeepEns** |
|---|---|---|
| Top-1 Accuracy | $\mathbf{85.17 \pm 0.16}\%$ | $84.16 \pm 0.00\%$ |
| Average Model Accuracy | $\mathbf{80.50 \pm 0.12}\%$ | $79.54 \pm 0.00\%$ |
| NLL | $\mathbf{0.6920 \pm 0.0070}$ | $0.7347 \pm 0.0000$ |
| Oracle NLL | $\mathbf{0.4027 \pm 0.0043}$ | $0.4408 \pm 0.0000$ |
| Brier Score | $\mathbf{0.2339 \pm 0.0018}$ | $0.2468 \pm 0.0000$ |
| ECE | $0.1400 \pm 0.0037$ | $\mathbf{0.1352 \pm 0.0000}$ |
| Ambiguity | $\mathbf{0.0466 \pm 0.0014}$ | $0.0462 \pm 0.0000$ |
| Normalized Disagreement | $0.2055 \pm 0.0021$ | $\mathbf{0.2069 \pm 0.0000}$ |
| Predictive Disagreement | $0.4110 \pm 0.0042$ | $\mathbf{0.4137 \pm 0.0000}$ |
| Number of models | 5 | 5 |
| Total samples | 10000 | 10000 |

## References

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

[2] Lars Kai Hansen and Peter Salamon. Neural network ensembles. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 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 M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems 34*, pages 7898–7911, 2021.

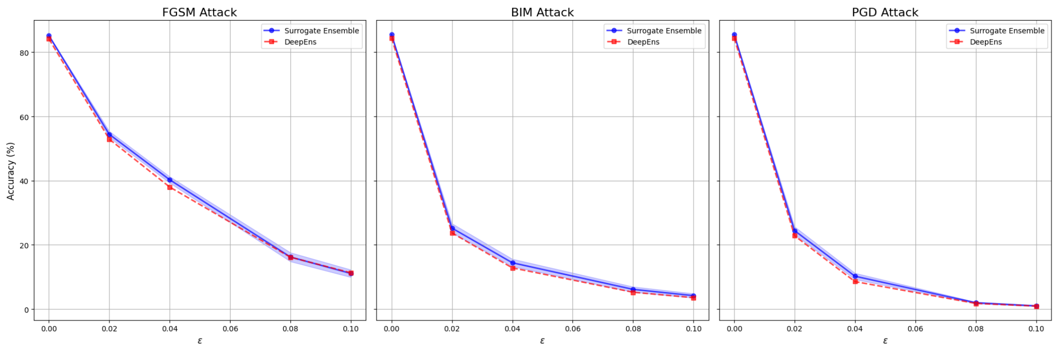

**Fig. 5:** CIFAR100: Ensemble accuracy under adversarial attacks (FGSM, BIM, PGD) for Surrogate Ensemble (blue) and DeepEns (red).

[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 D. Precup and Y. W. Teh, editors, *ICML*, volume 70, pages 2902–2911. PMLR, 2017.

[5] Esteban Real, Alok Aggarwal, Yanping Huang, and Quoc V. Le. Regularized evolution for image classifier architecture search. In *AAAI*, volume 33, pages 4780–4789. AAAI Press, 2019.

[6] Barret Zoph and Quoc V. Le. Neural architecture search with reinforcement learning. In *ICLR*. OpenReview.net, 2017.

[7] Sirui Xie, Hehui Zheng, Chunxiao Liu, and Liang Lin. Snas: Stochastic neural architecture search. In *ICLR*. OpenReview.net, 2019.

[8] Yuqiao Liu, Yanan Sun, Bing Xue, Mengjie Zhang, Gary G. Yen, and Kay Chen Tan. A survey on evolutionary neural architecture search. *IEEE Transactions on Neural Networks and Learning Systems*, 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 *KDD*, pages 1946–1956. ACM, 2019. 10.1145/3292500.3330648.

[10] Kirthevasan Kandasamy, Willie Neiswanger, Jeff Schneider, Barnabas Poczos, and Eric P. Xing. Neural architecture search with bayesian optimisation and optimal transport. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *NeurIPS*, volume 31, pages 201–211, 2018.

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

[12] Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *NeurIPS*, volume 30, pages 6405–6416, 2017.

[13] Yao Shu, Yizhou Chen, Zhongxiang Dai, and Bryan Kian Hsiang Low. Neural ensemble search via bayesian sampling. In C. C. Aggarwal and Z. Zhou, editors, *UAI*, volume 180, pages 1803–1812, 2022.

[14] Minghao Chen, Jianlong Fu, and Haibin Ling. One-shot neural ensemble architecture search by diversity-guided search space shrinking. In *CVPR*, pages 16525–16534, 2021. 10.1109/CVPR46437.2021.01626.

[15] 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.

[16] Zhichao Lu, Kalyanmoy Deb, Erik D. Goodman, Wolfgang Banzhaf, and Vishnu Naresh Boddeti. Nsganetv2: Evolutionary multi-objective surrogate-assisted neural architecture search. In A. Vedaldi, H. Bischof, T. Brox, and J. Frahm, editors, *ECCV 2020*, pages 35–51. Springer, 2020. 10.1007/978-3-030-58452-8_3.

[17] 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. *Artificial Intelligence in Medicine*, 119: 102154, 2021. 10.1016/j.artmed.2021.102154.

[18] Yu Xue, Zhenman Zhang, and Ferrante Neri. 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.

[19] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. *CoRR*, abs/1710.10903, 2017. 10.48550/arxiv.1710.10903.

[20] Wei Wen, Hanxiao Liu, Yiran Chen, Hai Li, Gabriel Bender, and Pieter-Jan Kindermans. Neural predictor for neural architecture search. In A. Vedaldi, H. Bischof, T. Brox, and J. Frahm, editors, *ECCV*, pages 660–676. Springer, 2020. 10.1007/978-3-030-58580-8_39.

[21] Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. In *CVPR*, pages 815–823, 2015. 10.1109/CVPR.2015.7298682.

[22] Hanxiao Liu, Karen Simonyan, and Yiming Yang. Darts: Differentiable architecture search. In *ICLR*. OpenReview.net, 2018.

[23] Xuanyi Dong and Yi Yang. Nas-bench-201: Extending the scope of reproducible neural architecture search. *CoRR*, 2020. 10.48550/arXiv.2001.00326.

[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.

[25] Liam Li and Ameet Talwalkar. Random search and reproducibility for neural architecture search. In J. Peters and D. Sontag, editors, *UAI*, volume 124, pages 367–377. PMLR, 2020.

## A NAS-Bench formats

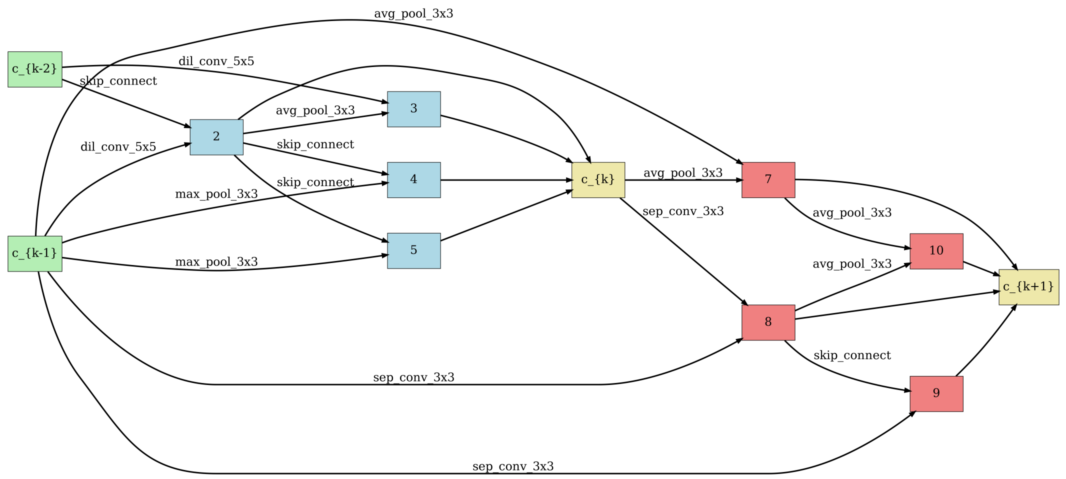

**Fig. 6:** Combined normal and reduced cells. The red vertices belong to the reduction cell; the blue vertices belong to the normal cell.

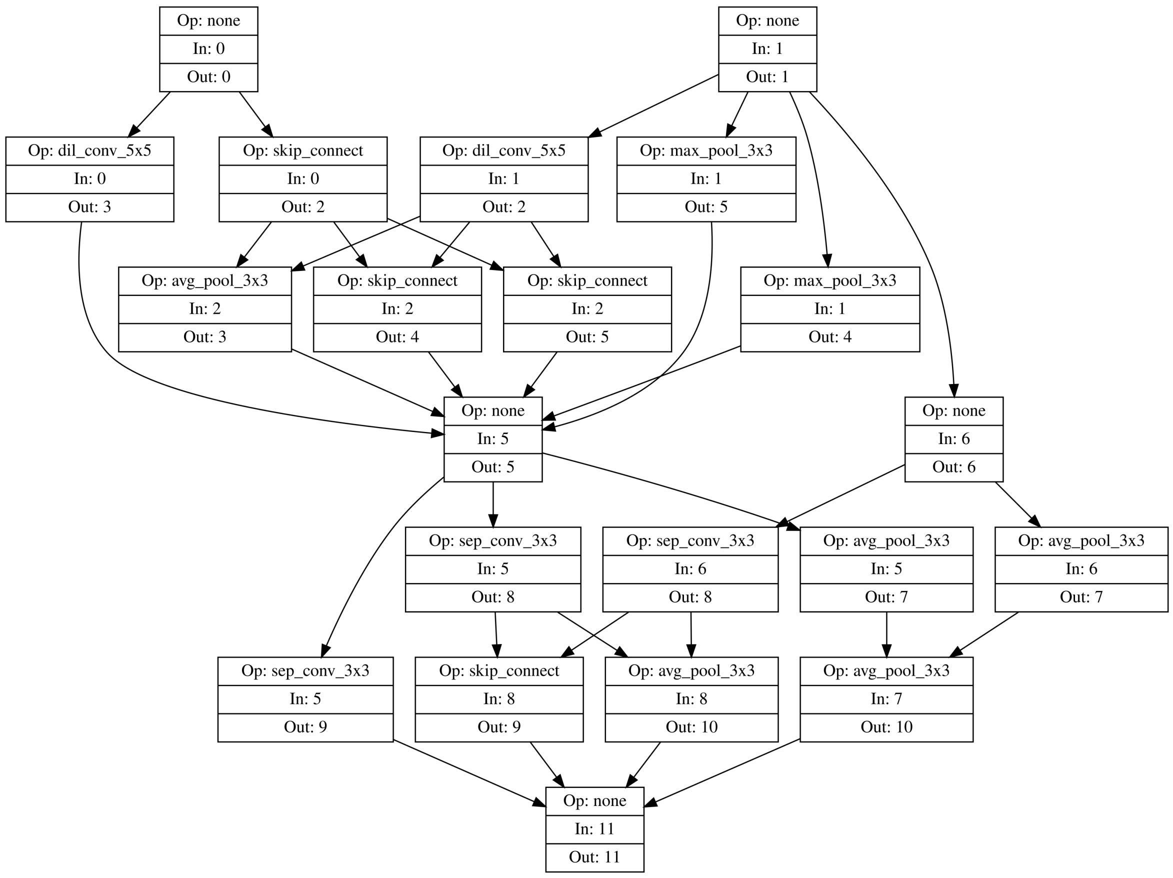

**Fig. 7:** Conversion of an architecture to the NAS-Bench-101 format.

## B Distributions of models in dataset

The main distributions of trained models you can see on Figure 8 and Figure 9:

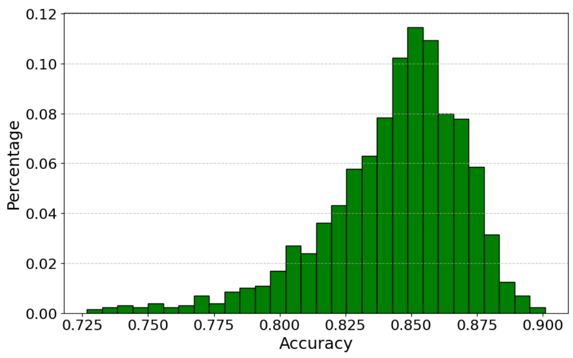

**Fig. 8:** Distribution of model accuracies

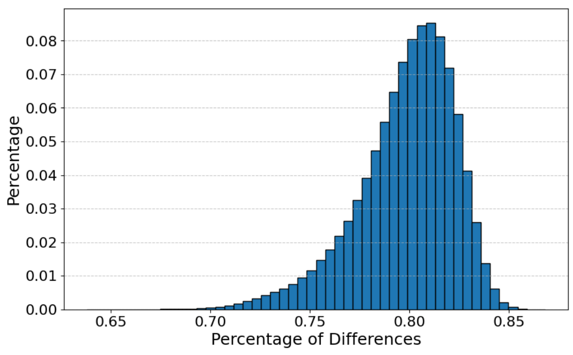

**Fig. 9:** Distribution of inter-model diversity

## C Training curve for surrogate funcitons

You can see the process of training accuracy surrogate function on Figure 11 and simularity surrogate function on Figure 10

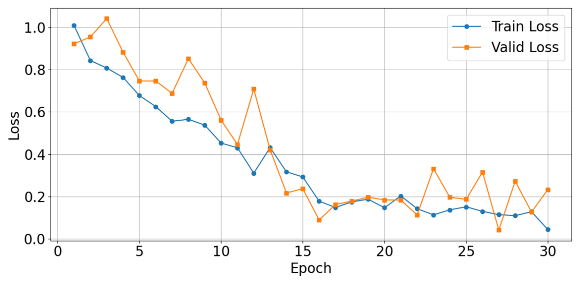

**Fig. 10:** Training curve of the surrogate model for diversity

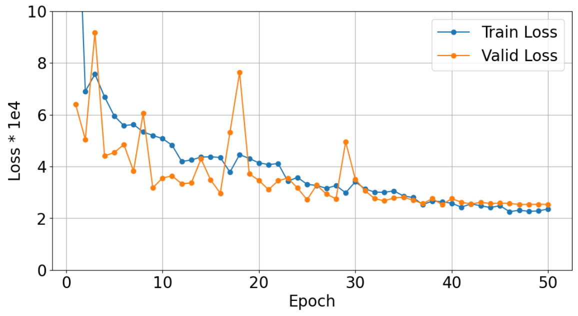

**Fig. 11:** Training curve of the surrogate model for accuracy