Diversity in Neural Architecture Search

Wenzheng Hu[1,3], Mingyang Li[1], Changhe Yuan[2],Changshui Zhang[1] and Jianqiang Wang[3]
[1]*The Insitute for Artificail Intelligence Tsinghua University (THUAI)*
Stat Key Lab of Intelligent Technologies and Systems
Beijing, National Research Center for Infomation Science and Technology (BNRist)
Department of Automation, Tsinghua University
Beijing, China
[2]*City University New York (CUNY) Queens College*
New York, USA
[3]*The State Key Laboratory of Automotive Satefy and Energy, Tsinghua University*
Beijing,China
Email:{hwz,zcs}@mail.tsinghua.edu.cn,limy17@mails.tsinghua.edu.cn,changhe.yuan@qc.cuny.edu,wjqlws@tsinghua.edu.cn

***Abstract*—Neural architecture search (NAS) is usually divided into two phases: *model search*, where candidate architectures go through an early training for a small number of epochs (e.g., 20) and a search strategy is used to find one or multiple top candidates, and *model tuning*, where the top candidates are trained fully (e.g., for 600 epochs) and one final best architecture is chosen. The top M-best strategy (M-Best) is typically used to help find better candidates during model search. However, the top M best solutions may concentrate in narrow similar areas and do not have enough diversity. Furthermore, empirical evidence suggests that performance distribution of the models which only go through the early training does not have a strong correlation with that of the models trained fully. Therefore, many of the M best solutions may turn out to be sub-optimal simultaneously because of their similarity, which limits the ability to find true top architectures. To alleviate the problems, we define *diverse M-best* architectures that are both of high quality and sufficiently different from each other based on a *novel* graph-based architecture distance. The concept is very general and is applicable to existing architecture search methods using top M-Best. To the best of our knowledge, this is the first time that diversity is introduced into architecture search. We applied the method in the progressive neural architecture search (PNAS) algorithm (Liu et al. 2018a). Experimental results show that our diverse M-Best is indeed beneficial for finding better architectures.**

***Index Terms*—diversity, architecture search, deep learning**

I. Introduction

Neural networks have been proven effective at solving many difficult problems, but designing good architectures can be challenging. Recently, many neural architecture search methods have been proposed to find effective architectures for convolution neural networks and recurrent neural networks [1]–[5]. Due to time complexity, many of these methods do an early training for candidate architectures with just a few epochs (e.g 20) during model search, and then perform full training (e.g. for 600 epochs) only for the top candidate. However, empirical evidence suggests that the performance distribution of the models after early training is different from that of the models trained fully. Even if we can perform optimal inference during model search, the best solution might be sub-optimal after fully trained. One way to mitigate this problem is to produce M best solutions. However, as shown in Fig. 1, top M best solutions found after early training may not perform well in the end. Furthermore, the M best architectures often concentrate in narrow areas and tend to be similar to the top solution. When one of these models fail, many other models fail simultaneously. This is an undesirable property.

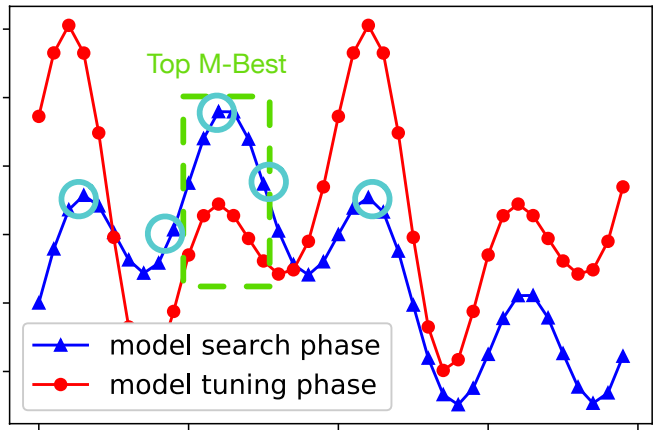


Fig. 1. Motivation for diverse M-Best: M best solutions found during model search within the green area do not perform well in the end, while diverse M best solutions may contain the solution performing really well, such as solutions in cyan circles. The x-axis represents some space distribution, and the y-axis represents the accuracy.

Different from top M-Best, we aim to find a set of solutions that are both of high quality and are qualitatively different from each other. Such a solution set has been proven effective in many other works [6]–[8], but it has not been considered in NAS before. There are two main challenges in finding such a solution set: one is how to evaluate the distance between different architectures, and the other is how to enforce diversity during architecture search.

A neural network can be viewed as a directed graph [9]. To evaluate the similarity, a natural way is to consider network alignment, which identifies similar regions of networks. How-

978-1-7281-6926-2/20/$31.00 ©2020 IEEE

ever, different from traditional network alignment, we concern about a scalar distance in diverse M-Best. In this paper, we borrow ideas from network alignment and propose a method to select diverse M-best architectures. We make the following contributions:

- We introduce a graph based method to evaluate similarity as well as distance between two neural network architectures.
- We introduce diverse M-Best into architecture search. Experiments confirm the effectiveness of diversity. To the best of our knowledge, this is also the first time that diverse M-Best is introduced into architecture search.

II. Related Work

A. Architecture Search

There has been a recent surge of interest in generating neural network architecture automatically. A popular direction is reinforcement learning that generally utilize a recurrent neural network (RNN) to design the network architecture, e.g. [2] uses a RNN as controller to sequentially generate coded architectures, and the RNN is trained to maximize the expected accuracy of the generated architectures on a validation set. Several further improved methods were developed based on this idea, including progressive search [4] and parameter share [9]. There are also some other methods that use RNNs to generate architectures, such as [3], [10]. Yet another heavily investigated approach is using evolutionary algorithm to explore the architecture space, such as [5], [11], [12]. More recently, gradient based methods have been developed. [13] developed a Bayesian optimization method for architecture search. [14] selects connections for neurons using a softmax classifier.

B. Network Alignment

Network alignment forms a matching or alignment between two graphs by identifying similar regions of networks. It is deeply intertwined with many classical computational problems, such as graph isomorphism, maximum common subgraph and quadratic assignment. Usually, network alignment solves an optimization problem, e.g minimizing $||A - PBP^T||_F$, where A and B are the adjacency matrices of two respective networks [15].

A classic approach is IsoRank algorithm [16], which propagates the pairwise node similarity in the Kronecker product graph. Then, several works extend the algorithm. For example, [17] extends the algorithm to align multiple networks; [18] proposes a fast and scalable network alignment by uncoupling and decomposition; and [19] uses sequence similarity and matrix tri-factorization to incorporate network structure into alignment.

[20] formulates network alignment as an integer quadratic programming problem. [21] utilizes the alternating projected gradient descent to solve a bipartite network alignment problem. [22] uses prior information about possible relationships and generates the concept of the facility location. [23] and [24] explore fast alignment on attribute networks.

III. Architecture Diversity

There are two technical challenges in order to introduce diversity into neural architecture search. One is how to define *architecture distance*, i.e., how to evaluate the distance between different architectures. The other is how to select diverse architectures. We tackle the challenges in the following.

A. Architecture Distance

Since architectures can be viewed as directed graphs, a natural way is to consider network alignment, which identifies similar regions of networks. However, different from traditional network alignment, we need a scalar distance. Borrowing ideas from network alignments, we propose an architecture distance as follow.

In this section we introduce an attribute-based architecture distance. Let $G_1(\mathcal{V}_1, \mathcal{E}_1)$ and $G_2(\mathcal{V}_2, \mathcal{E}_2)$ be two graphs with node sets $\mathcal{V}_1$, $\mathcal{V}_2$ and edge sets $\mathcal{E}_1$, $\mathcal{E}_2$ representing two neural architectures ϕ_1 and ϕ_2 respectively. In an architecture, nodes represent operations, and edges represent dataflows. Building on existing research on graph alignment [15], [23], [24], we define architecture distance based on both dataflows and operations.

1) Dataflow Representation: In graph alignment, the basic assumption is that aligned nodes have similar connections or degrees. Here we consider both the node's in-degrees and out-degrees. For a node v in $\mathcal{V}$ in graph G, $\mathcal{N}_v^{k-}$ is the node set within k-steps from v in the upstream graph, and $\mathcal{N}_v^{k+}$ is the node set within k-steps in the downstream graph. We define the k-step neighborhood $\mathcal{N}_v^k$, where $\mathcal{N}_v^k = \mathcal{N}_v^{k-} \bigcup \mathcal{N}_v^{k+}$, to capture the degree information of node v. We store the in-degree information and out-degree information in vectors d_v^{k-} and d_v^{k+} respectively, where the i-th entry of d_v^{k-} is the number of nodes in the k-step neighborhood with in-degree i, and the i-th entry of d_v^{k+} is the number of nodes in the k-step neighborhood with out-degree i. To capture a regional feature, we define the dataflow representation of node v with K different level neighborhoods as follows,

$$d_v^- = \sum_{k=1}^{K} \gamma^{k-1} d_v^{k-}, d_v^+ = \sum_{k=1}^{K} \gamma^{k-1} d_v^{k+}, \tag{1}$$

where $\gamma \in (0, 1)$ is the attenuation factor controlling the influence of different neighborhoods.

2) Operation Representation: In an architecture graph G, different nodes stand for different operations. They may come from different types of layers, e.g., convolution, pooling or concatenating. Even if they have the same type, they may be different in other aspects, e.g. number of output channels or kernel size. To account for these factors, we extract features for nodes. For a node v in $\mathcal{V}$, f_v is the feature vector, with the i-th entry of f_v (namely f_v^i) corresponding to the value of the i-th attribute. We consider several basic but important attributes for the node representation, including type of layer, channels, kernel size, stride and padding. Note that features of different nodes have the same length. If a node does not have certain attribute (e.g. a concatenating layer does not have

kernel sizes), we set the corresponding value to a default value (say 0.0).

3) Node Similarity: Based on the above representations of operations and dataflows, we define the following similarity function to compare two nodes (v_1 and v_2) across different graphs:

$$s(v_1, v_2) = exp(-\gamma_{in} \cdot d_{in} - \gamma_{out} \cdot d_{out} - \gamma_f \cdot d_f), \quad (2)$$

$$d_{in} = ||d^-_{v_1} - d^-_{v_2}||_2^2, \quad (3)$$

$$d_{out} = ||d^+_{v_1} - d^+_{v_2}||_2^2, \quad (4)$$

$$d_f = \frac{1}{|A_0|} \sum_{i \in A_0} \mathbb{1}(f^i_{v_1} \neq f^i_{v_2}) + D(f^{\overline{A}_0}_{v_1}, f^{\overline{A}_0}_{v_2}) \quad (5)$$

where γ_{in}, γ_{out} and γ_f are constants controlling the effect of dataflow representations and operation features, A_0 is an index set of categorical features, $f^{\overline{A}_0}_{v_1}$ and $f^{\overline{A}_0}_{v_2}$ are feature vectors removing corresponding values in set A_0, and $D(\cdot)$ can be Minkowski distance or Cosine distance. We use Cosine distance in our experiments. Such a similarity function has a comprehensive coverage of both connections and operations.

4) Graph Distance: To calculate the distance of graphs $G_1(\mathcal{V}_1, \mathcal{E}_1)$ and $G_2(\mathcal{V}_2, \mathcal{E}_2)$, a common approach is to make use of the similarity matrix S of all pairs of nodes in both graphs, which takes $\Theta(n^2)$ time.

In network alignment, some existing approaches factorize the similarity matrix S, embed nodes (e.g by minimizing $||S - FZ^\top||_F^2$) and align nodes using embedding features. Following existing research on network alignment [23], [24], we also calculate distance in an embedding space. Due to the high complexity of graphs and huge number of possible architectures, we reduce the complexity of similarity matrix computation by using SVD in computing the embedding features. Our approach is based on the Nystrom method [25]. The method chooses c columns from a matrix M uniformly at random, and constructs an approximation of the form $\tilde{M} = CW^\dagger C^\top$, where C is the $n \times c$ matrix consisting of the c chosen columns and $W^\dagger$ is the pseudo-inverse of matrix that consists of the intersection of those c columns with the corresponding c rows. To alleviate uncertainty introduced by the random selection when calculating distance and diversity, we extend the Nystrom method with fixed landmark nodes. From our point of view, nodes with many connections, namely dataflow hubs, are important nodes. We therefore select top p nodes with the highest degrees as landmarks. For an architecture graph $G(\mathcal{V}, \mathcal{E})$, we empirically choose $p = \log_2(|\mathcal{V}|)$ landmark nodes in our experiments. Let Γ_1 and Γ_2 be the landmark sets of $G_1(\mathcal{V}_1, \mathcal{E}_1)$ and $G_2(\mathcal{V}_2, \mathcal{E}_2)$ respectively. S_Γ is the similarity matrix of landmark node set Γ, where $\Gamma = \Gamma_1 \bigcup \Gamma_2$ and

$$S_\Gamma(i, j) = s(v_i, v_j), \forall i, j \in [0, |\Gamma|), v_i \in \Gamma, v_j \in \Gamma. \quad (6)$$

Then the similarity matrix $S \in \mathbb{R}^{n \times n}$ ($n = |\mathcal{V}_1 + \mathcal{V}_2|$) of each graphs is approximated by $S \approx CS^\dagger_\Gamma C^\top$, where $S^\dagger_\Gamma$ is the pseudo-inverse of S_Γ and $C \in \mathbb{R}^{n \times p}$. Let $S^\dagger_\Gamma = U \Sigma V^\top$, the singular value decomposition of $S^\dagger_\Gamma$. We have

$$S \approx CS^\dagger_\Gamma C^\top = (CU\Sigma^{\frac{1}{2}}) \cdot (CV\Sigma^{\frac{1}{2}})^\top. \quad (7)$$

Let $F = CU\Sigma^{\frac{1}{2}}$. It can be simply regarded as a solution of minimizing $||S - FZ^\top||_F^2$. So we respectively get the embedding features of nodes in G_1 and G_2 as follows,

$$F_1 = S_1 U \Sigma^{\frac{1}{2}}, \quad S_1 \in \mathbb{R}^{|\mathcal{V}_1| \times p}, \quad (8)$$

$$F_2 = S_2 U \Sigma^{\frac{1}{2}}, \quad S_2 \in \mathbb{R}^{|\mathcal{V}_2| \times p}, \quad (9)$$

where S_1 is the similarity matrix between G_1 and Γ, S_2 is the similarity matrix between G_2 and Γ. In the embedding space, we define distance between G_1 and G_2 as follow,

$$\Delta(G_1, G_2) = \frac{1}{|\mathcal{V}_1|} \sum_{i=1}^{|\mathcal{V}_1|} \min_j(D_{ij}) + \frac{1}{|\mathcal{V}_2|} \sum_{j=1}^{|\mathcal{V}_2|} \min_i(D_{ij}), \quad (10)$$

where $D_{ij} = ||F_1(i) - F_2(j)||, D \in \mathbb{R}^{|\mathcal{V}_1| \times |\mathcal{V}_2|}$.

Algorithm 1 Architecture Distance

Require:
Architecture $G_1(\mathcal{V}_1, \mathcal{E}_1)$ and $G_2(\mathcal{V}_2, \mathcal{E}_2)$;
Neighborhood level K;

Ensure:
$\mathcal{V} = \mathcal{V}_1 \bigcup \mathcal{V}_2$;
for node v in $\mathcal{V}$ **do**
  compute dataflow representations d^-_v, d^+_v;
  extract operation features f_v;
end for
choose landmark node set $\Gamma = \Gamma_1 \bigcup \Gamma_2$;
compute the full similarity matrix S_Γ for Γ;
$[U, \Sigma, V] = SVD(S^\dagger_\Gamma)$;
compute the similarity matrix S_1 and S_2;
$F_1 = S_1 U \Sigma^{\frac{1}{2}}$,$F_2 = S_2 U \Sigma^{\frac{1}{2}}$;
$dist = \frac{1}{|\mathcal{V}_1|} \sum_{i=1}^{|\mathcal{V}_1|} \min_j(D_{ij}) + \frac{1}{|\mathcal{V}_2|} \sum_{j=1}^{|\mathcal{V}_2|} \min_i(D_{ij})$;
return $dist$

The time complexity of computing the architecture distance is $O(np)$, where n is the number of nodes and p is the number of landmark nodes ($p \ll n$). Generally, compared to the whole architecture search, the overhead is typically negligible.

B. Diverse M-Best

In this section, we introduce a method called diverse M-Best in the context of architecture search. M-Best is the problem of finding the M architectures with highest potential. For example, RL-based TreeCell [10] finds top 10 candidate cells discovered in the search; the evaluation method [11] finds top 1000 individuals; and PNAS [4] finds top 256 architectures in the search.

Given a scale parameter δ, we define the δ-architectures as follows.

Definition 1: (δ-architectures). The δ-neighborhood of an architecture ϕ, $\mathcal{N}_\delta(\phi)$, is the ball centered at architecture ϕ with radius δ, formally, $\{s|\Delta(s, \phi) \leq \delta\}$. An architecture is a δ-architecture if and only if its performance is higher than all other elements in $\mathcal{N}_\delta(\phi)$.

Let $\rho(\phi) : \Psi \to \mathbb{R}$ be function defining the performance of architecture ϕ, where Ψ is the architecture space. We measure the performance of an architecture using accuracy, and aim

to find the architecture with the highest maximum a performance (MAP). In most of search methods, the performance mentioned here is the accuracy on validation set.

1) General Solution: With the key concept defined, we now describe the proposed diverse M-Best method. Recall that the goal is to produce a diverse set of solutions. We introduce an iterative algorithm to solve this problem, where the next best solution is defined as the MAP at least some minimum distance away from current MAPs. Let $\{\phi_1, ..., \phi_t\}$ denote the incumbent solutions, and let us search for the next solution. We propose the following general formulation:

$$\begin{aligned} \max_{\phi_{t+1} \in \Psi} \quad & \rho(\phi_{t+1}) \\ s.t \quad & \phi_{t+1} \in \overline{\mathcal{N}}_\delta(\phi_i), \forall i \in \{1, 2, ..., t\}, \end{aligned} \tag{11}$$

where $\overline{\mathcal{N}}_\delta(\phi_i)$ is the complementary set of $\mathcal{N}_\delta(\phi_i)$. Intuitively, we can see that the above formulation searches for the solution that are at least δ away from the current solutions. The top M-Best can be seen as a special case of this formulation, where $\Delta(\cdot, \cdot)$ is a 0-1 similarity (e.g $\Delta(G_1, G_2) = \mathbb{1}(G_1 == G_2)$) and $\delta = 0$. Thus the formulation simply select the next MAP.

2) Relaxation: The general version of diverse M-Best is suitable only for known distributions. Usually we do not know the distribution exactly. We therefore define the following relaxed version of diverse M-Best:

$$\begin{aligned} \max_{\phi_{t+1} \in \Psi} \quad & \rho(\phi_{t+1}) \\ s.t \quad & \phi_{t+1} \notin_p \mathcal{N}_\delta(\phi_i), \forall i \in \{1, 2, ..., t\}, \end{aligned} \tag{12}$$

where $\notin_p$ indicates ϕ_{t+1} does not belong to $\mathcal{N}_\delta(\phi_i)$ with probability p. The general formulation above can be seen as a special case of the relaxation formulation with $p = 1$.

IV. Experiments

In this section, we present experimental evaluations of the effectiveness of diversity in neural network architecture search. We first study the relation between the performance distribution of models trained for just a few epochs and those trained fully on the CIFAR-10 dataset to motivate the necessity of diversity. We then present accuracy results of diverse M-Best on the classification tasks on CIFAR-10 and ImageNet. Finally, we study the sensitivity of hyper-parameters in the diverse M-Best search.

A. Experimental Details

We conducted most of our experiments on the popular CIFAR-10 dataset except the classification experiments that are on both CIFAR-10 and ImageNet. It contains 60,000 images consisting of 32×32 pixels, including 50,000 training images and 10,000 test images in 10 different classes. During our architecture search, we randomly sample 5,000 images from the training set without replacement as a validation set. All images are processed by the standard data pre-processing and augmentation techniques, e.g., subtracting the mean, dividing the standard deviation, cropping the 32×32 patches from images up-sampled to 40×40 and randomly flipping horizontally.

TABLE I

SUBSPACE OF PNAS-LIKE ARCHITECTURES FOR OUR RANDOMLY SAMPLING, WHERE N IS THE NUMBER OF CELLS, F IS THE NUMBER OF FILTERS, H^{c-1} IS THE OUTPUT OF PREVIOUS CELL, H^{c-2} IS THE OUTPUT OF PREVIOUS-PREVIOUS CELL AND $b[: c-1]$ ARE THE OUTPUTS OF PREVIOUS BLOCKS IN THE SAME CELL.

Parameters	Values
N	$2 \sim 5$
F	$20 \sim 50$
Operations	[conv1x1,sep3x3,sep5x5,sep7x7, avg pool3x3,max pool3x3,iden,dil3x3.]
Connection	$H^{c-1}, H^{c-2}, b[: c-1]$

We follow the setting in PNAS [4]. The maximum number of epochs used on CIFAR-10 is 600. During full-step training, we use auxiliary classifier located at $2/3$ of the total number of cells with 0.4 loss weight. Probability of dropping is 0.4 and clipping gradient exceeding 5 is used for regularization. We train models using a learning rate of 0.025 with cosine decay.

B. Distribution Exploration

In neural architecture search, researchers usually train models for a small number of epochs, evaluate these models on validation set, pick up the best one or M and then take a full training for the selected top candidates. It builds on the implicit assumption that performances of models after early training are strongly correlated to those trained fully. In this section, we randomly select 100 models (with different connections, operations and depths) from a subspace of PNAS-like architecture for 5 times, which removes some extreme hyperparameter values and is described in Table I . Then we display the relationship between their performances after being trained for just a few epochs and after being fully trained. What we reported is the average performance.

Fig 2(a) displays the scatter plot of sampled architectures. The x-axis are their ranks by validation error after being trained for 20 epochs (search phase), and the y-axis are their ranks after being fully trained for 600 epochs (tuning phase). The correlation coefficient of these points is 0.347. It is hard to argue that an architecture performing well with a small number of epochs will likely remain strong after being fully trained. Otherwise, points in Fig 2(a) should locate on the line y=x.In Fig 2(b), we can see that the correlation coefficient increases with training epochs. But even if the number of epochs increased to 100, the correlation coefficient is only about 0.5. Further more, researchers usually take an epoch far less than 100. Fig 2(c) shows the probability of successfully hitting the real best architecture from the early training (less than 100) distribution, which performs best after full training. The x-axis k is the number of selected top M-Best architectures. We can see that diversity indeed improved the probability of successfully selecting the best architecture during the early training. The gap between diverse M-Best and simple M-Best seems to enlarge with k as well. Fig 2(d)

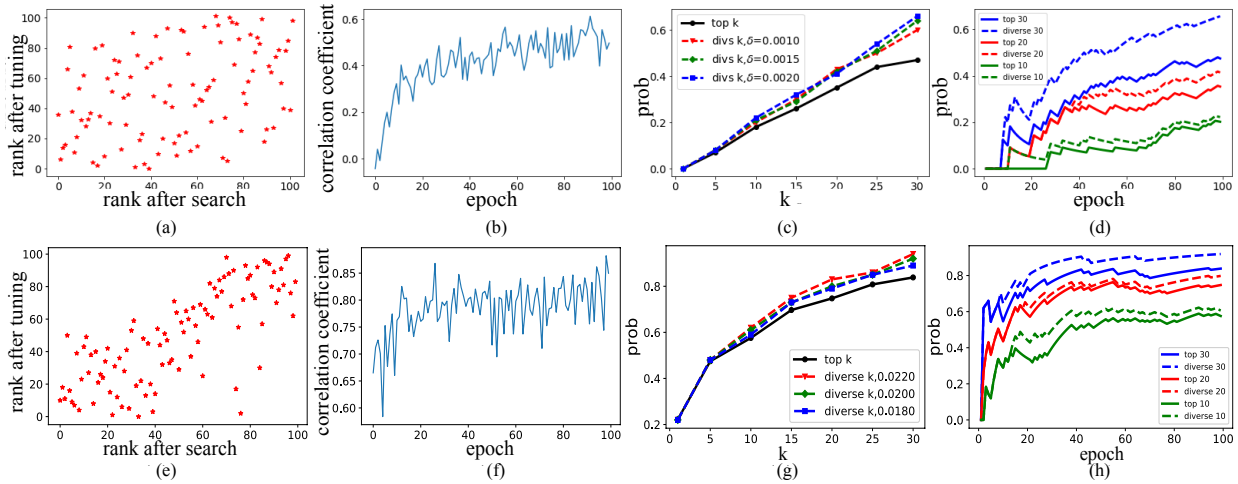


Fig. 2. $(a) \sim (d)$ depict PNAS-like architectures,$(e) \sim (h)$ depict Dense-like architectures. (a) and (e) display the scatter plot of random sampled architectures. The x-axis is their validation ranks after being trained for 20 epochs (*model search*), and the y-axis the rank after being fully trained (*model tuning*). (b) and (f) display the relationship between correlation coefficients and epoch. (c) and (g) display the probability of hitting the best architecture with less than 100 epoch training.(d) and (h) display the probability of hitting the best architectures within different epochs.

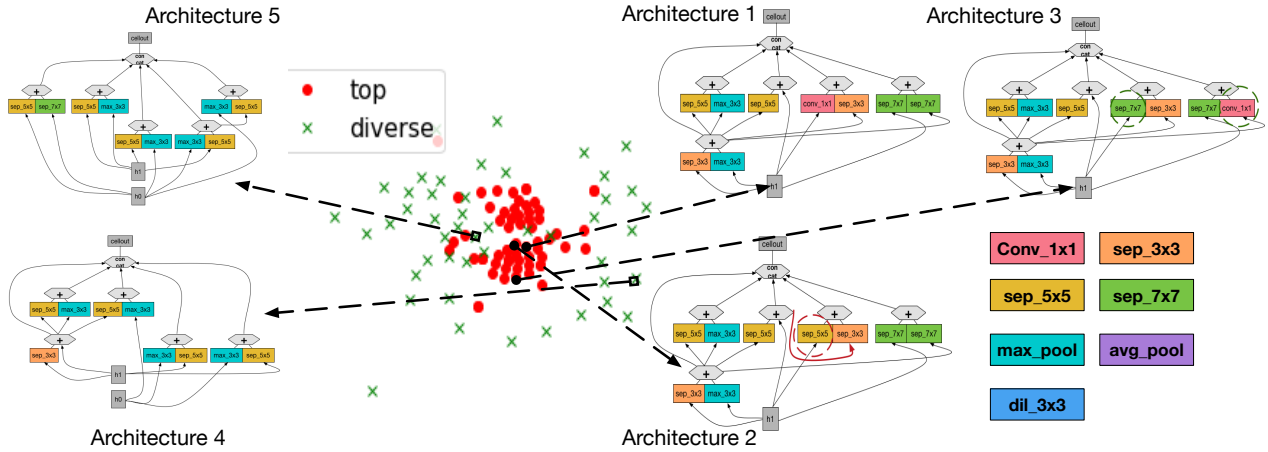


Fig. 3. Visualizations of architectures selected by PNAS and Divs-PNAS. Red circles are architectures by PNAS which uses top M-Best. Green crosses are from Divs-PNAS that uses diverse M-Best.

illustrates the probability of successfully selecting the best architecture within different numbers of epochs in the early training. Compared to top M-Best, it is clear that diverse M-Best improved the probability.

Similarly, we do an experiment in a Dense-like architecture space. Results shown in Fig 2 $(e) \sim (h)$ also present that architecture which performs well after fully trained may not perform well with only a small number of epochs.

C. Diverse Search

Diversity not only improves the probability of hitting the best architecture but also the overall quality of the solution set. To illustrate that, we select the PNAS method [4], which keeps M Best solutions during its search, as the baseline. PNAS sequentially picks up top M cells and then expands the selected cells by increasing the number of blocks. There are 5.6×10^{14} possible cell architectures in the search space. We simply replace M-Best with our diverse M-Best method. We name our architecture search method as Divs-PNAS.

TABLE II
STUDENT'S T HYPOTHESIS TEST FOR THE MEANS OF DIVS-PNAS AND PNAS. $H_0 : \mu_{pnas} \geq \mu_{divspans}$.

	#Block3	#Block4	#Block5
P-value	0.026	0.010	0.0001

Following settings used in PNAS, we fitted an ensemble of 5 LSTM predictors and trained CNNs for 20 epochs during the architecture search. We ran both PNAS and Divs-PNAS

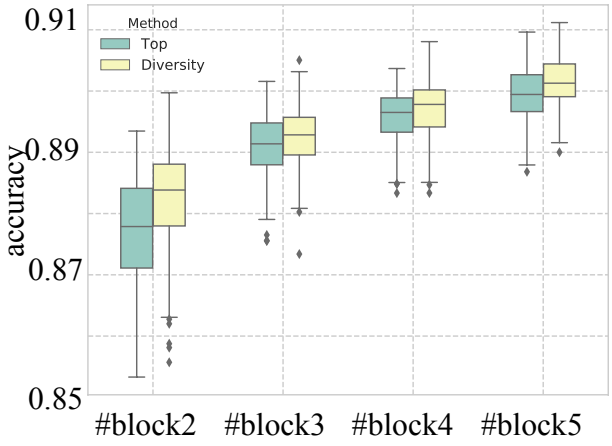


Fig. 4. A Box-plot of of diverse strategy and top strategy on Cifar10.

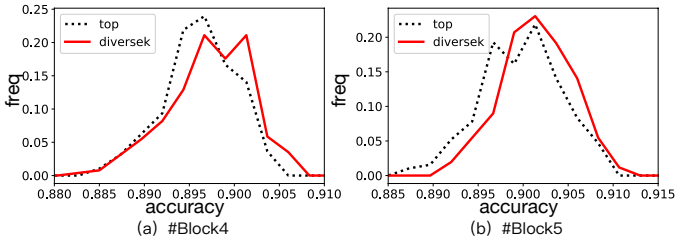


Fig. 5. (a) Frequency charts of PNAS and Divs-PNAS with #Block=4.(b) Frequency charts of PNAS and Divs-PNAS with #Block=5.

for a maximal number of blocks being 5. Considering our limited computational resources, we only kept top 64 (diverse) mode solutions instead of 256 modes as in [4]. To overcome the difference introduced by predictors, we use the relaxation version of diverse M-Best by fixing the relaxing probability to be 0.5. We repeated the experiments 5 times and reported the average performance.

We provide several concrete examples of the difference between PNAS and Divs-PNAS. Fig 3 provides a spatial distribution of architectures with 5 blocks drawn with t-SNE. Red dots are top M architectures selected by PNAS, while green crosses are architectures found by Divs-PNAS. The proximity of the solutions represent graph distances between them. Some of these architectures are visualized. It is easy to see the architectures are similar. Architecture 1, 2 and 3 are three of the top M-Best architectures. However, Architecture 3 only differs slightly from Architecture 1 in merely two operations indicated by the green circles. Similarly, Architecture 2 only differs from Architecture 1 in merely one operation and one connection indicated by the red circle and arrow. In comparison, architectures selected by Divs-PNAS are much more different from each other.

Fig 4 presents the box-plot of our diverse strategy and the top strategy on Cifar-10. As illustrated in Fig 4, our diverse strategy generally picks up more high performance architectures than the top strategy. We calculate the Student's T-test for the means of Divs-PNAS and PNAS. P-values are

TABLE III

RESULTS ON CIFAR10.(A) PRESENTS THE COMPARISON OF PNAS AND DIVS-PNAS WITH DIFFERENT NUMBER OF BLOCKS. COLUMNS WITH "†" ARE DIRECTLY TAKEN FROM [4]. (B) GIVES SOME OTHER RECENT METHODS TO SHOW THE SIGNIFICANCE OF THE IMPROVEMENT BROUGHT BY DIVERSE STRATEGY.

			PNAS		Divs-PNAS	
#Block	**N†**	**F†**	**Error†**	**Params†**	**Error**	**Params**
3	6	32	3.70±0.12	1.8M	**3.49±0.08**	2.4M
4	4	66	3.50±0.10	3.0M	**3.15±0.09**	2.7M
5	3	48	3.41±0.09	3.2M	**3.05±0.07**	2.8M

(a)

Method	**Error**	**Params**
NAONet [27]	3.18	10.6M
DSO-NAS-full [28]	**2.95**	3.0M
AmoebaNet-A [5]	3.34	3.2M
DARTS(1st-order) [14]	3.00	3.3M
PNAS [4]	3.41	3.2M
Divs-PNASNet-5	3.05	**2.8M**

(b)

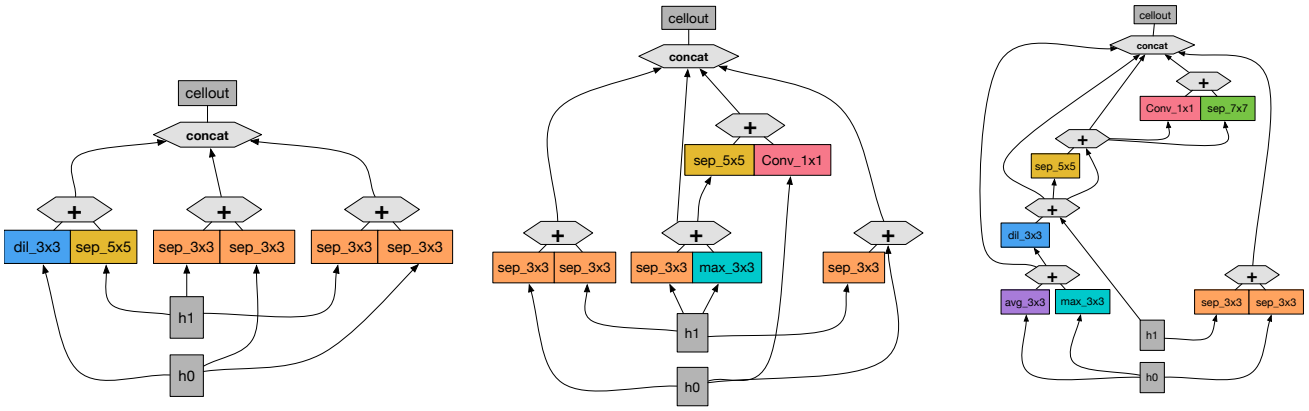
presented in Table II, which also indicate that architectures picked by Divs-PNAS are generally better than PNAS. Fig 5 is the frequency-plot of different number of blocks. From Fig 5, we can see that Divs-PNAS has a tendency to identify more architectures with better validation performance in comparison to PNAS. Moreover, the tendency is more clear with increasing number of blocks.

To illustrate the effectiveness of diversity strategy, we also do a simple ensemble experiment .In general, ensemble of diverse learners can get a better learner [26]. We simply take top 5 models and diverse 5 models in model search, and then we respectively do classification on CIFAR-10 by voting after model tuning. Compared to the ensemble of top models, ensemble of diverse models get 10.8% relative error reduction (top $3.7 \pm 0.22\%$ vs diverse $3.3 \pm 0.16\%$), which shows the diversity of models exists. Thus, to find a set of architectures are both of high quality and are qualitatively different from each other is significant.

D. Results on CIFAR-10 and ImageNet

We now discuss the final model selected by our Divs-PNAS, and compare it to the one selected by PNAS. We implement Divs-PNAS with K=256 (namely diverse 256-Best Modes) as the setting in PNAS. For simplicity, we used the same N and F optimized for PNAS after selecting the cell structure. The results of CIFAR10 are shown in Table III. In Table III (a), we see that diverse M-Best indeed improved the accuracy in comparison to top M-Best. When using the same number of blocks, Divs-PNAS reduced the relative error rate by 9% on average. Table III (b) presents a comparison of Divs-PNAS and some recent methods. Compared to the improvement of these recent results, we can see that the improvement bought by our diverse strategy is significant. Note that our diverse M-Best method is not applicable to algorithms that do use the M-Best strategy.

The best 5-block cell we discovered on CIFAR10, which we call Divs-PNASNet-5, is visualized in Fig 6. We applied the cell to ImageNet classification. We conducted experiment un-



(a) Divs-PNAS-3. (b) Divs-PNAS-4. (c) Divs-PNAS-5.

Fig. 6. (a)Visualize Divs-PNAS-3.(b)Visualize Divs-PNAS-4.(c)Visualize Divs-PNAS-5.

TABLE IV
IMAGENET RESULTS IN THE MOBILE SETTING.

Model	Params	Top1	Top5	Mult-Adds	Search Method
MobileNet-224 [29]	4.2M	70.6	89.5	569M	manual
ShuffleNet 2x(V1) [30]	$\sim$ 5M	73.7	89.8	524M	manual
NASNet-A(N=4,F=44) [2]	5.3M	74.0	91.6	564M	RL
AmoebaNet-A(N=4,F=50) [5]	5.1M	74.5	**92.0**	555M	evolution
AmoebaNet-B(N=3,F=62) [5]	5.3M	74.0	91.5	555M	evolution
DARTS [14]	4.9M	73.1	91.0	595M	gradient-based
PNASNet-5(N=3,F=54) [4]	5.1M	74.2	91.9	588M	SMBO
Divs-PNASNet-5(N=3,F=54)(ours)	**4.6M**	74.4	91.8	552M	SMBO
Divs-PNASNet-5(N=4,F=48)(ours)	4.8M	**74.7**	**92.0**	578M	SMBO

der the Mobile setting where the input image size is 224×224. The results are summarized in Table IV. Divs-PNASNet-5 achieved slightly better performance than PNASNet-5 with fewer parameters (74.4% top-1 accuracy for Divs-PNASNet-5 vs 74.2% for PNASNet-5) under the same N and F. However, the top-5 accuracy is 0.1% lower than PNASNet-5. Because we did not optimize N and F for Divs-PNAS and simply used the same combination optimized for PNAS. After optimization, we get a better Divs-PNASNet-5(74.7% top-1 accuracy, 92.0% top-5 accuracy) as shown in Table IV.Compared to the improvement of some other recent results, we can see that the improvement bought by the diverse strategy is significant.

E. Exploring the values of δ

An important parameter in defining diversity is the radius δ. Following the setting in Section Diverse Search IV-C, we ran Divs-PNAS on CIFAR10 with different δs. Experiments were repeated for 5 times, and we report the average values in Fig 7.

Given other parameters, as δ increases, different statistics (including quartile, median, best one-Mode, average of best 10-Modes) of architectures with 5 blocks show that the performance of diverse M-Best improved significantly. However, the performance started deteriorating when δ became too large. The reason is that a relatively small δ increases the available architecture space areas. However, as δ kept increasing, diverse M-Best may include more weak architectures. The optimal value of δ is highly domain dependent and should be tuned for optimal performance.

Given δ, as $\gamma_{in}, \gamma_{out}$ and γ_f change, the performance of best one has a dynamic range as show in Fig 7 (a). Fig 7 (b) shows the effect of γ_{in} (or γ_{out}, or γ_f) in comparison with a baseline parameter setting.

A practical way to obtain an appropriate setting is to do some simple experiments, such as searching for architectures with just only 2 blocks.

V. DISCUSSION AND FUTURE WORK

The main contributions of this work are two folds: (1) a novel graph based distance for architectures and (2) the diverse M-Best method in architecture search. In our experiments, we mainly tested our method with PNAS which uses top M-Best in the search process. Another possible baseline is the evolution-based algorithm, although we have not done such experiments due to the time cost. Since our proposed method is agnostic of the underlying search procedure, we believe our current findings should be equally applicable there.

Note that the major overhead of our method is only computing the architecture distances, and its time complexity is $O(np)$, where n is the number of nodes and p is the number of critical points in the architecture ($p \ll n$). In the Divs-PNAS, since we usually limit K to be relatively small, the total number of architecture distances we need to calculate is also quite reasonable. Compared to the whole architecture

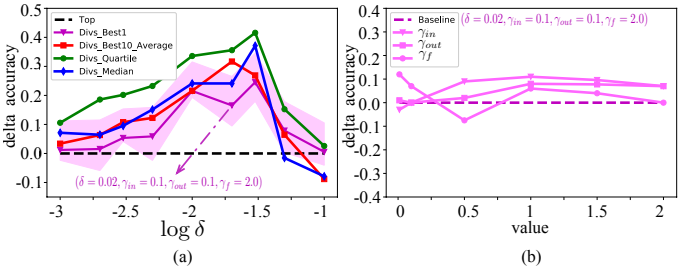


Fig. 7. (a)Improvement with delta. The x-axis is values of $\log \delta$, and the y-axis is the improvement compared to top M-Best Mode.(b) The effect of $\gamma_{in}, \gamma_{out}, \gamma_f$.

search, the overhead is typically negligible. In other words, the searching complexity of the algorithm remains in the same order when diversity is used.

There are many possible directions for future work. An interesting avenue of research is to investigate how to take an adaptive adjustment to the radius. An adaptive adjustment may make the method more effective. More node features may also contribute to the architecture distance and neural architecture search. More distances and how to introduce diverse strategy into methods without M-Best strategy are also interesting.

VI. Conclusion

In this paper, we do not aim to propose a brand new search algorithm, but to introduce the diverse M-Best concept into NAS. We empirically explore the relationship between the performance distribution of models trained for just a few epochs versus those trained fully. Results show they are not very strongly correlated. Then, we introduce a novel graph based method to evaluate the distance between two architectures, and introduce diverse M-Best into neural architecture search for the first time. Experiments show that diversity did help in improving the probability of finding the best architecture.

Acknowledgment

This work is (jointly or partly) funded by the NSFC(Grant No.61751308 and 61876095)the Beijing Natural Science Foundation(Grant No.L172037), and China Postdoctoral Science Foundation. The authors thank the support of Tsinghua-DiDi Joint Research Center for Future Mobility, and Dr. Yuan thanks the support of NSF grant IIS-1829560.

References

[1] B. Baker, O. Gupta, N. Naik, and R. Raskar, "Designing neural network architectures using reinforcement learning," in *The International Conference on Learning Representations (ICLR)*, 2017.

[2] B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le, "Learning transferable architectures for scalable image recognition," *arXiv preprint arXiv:1707.07012*, vol. 2, no. 6, 2017.

[3] H. Cai, T. Chen, W. Zhang, Y. Yu, and J. Wang, "Efficient architecture search by network transformation," in *The Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[4] C. Liu, B. Zoph, J. Shlens, W. Hua, L.-J. Li, L. Fei-Fei, A. Yuille, J. Huang, and K. Murphy, "Progressive neural architecture search," in *European Conference on Computer Vision*, 2018.

[5] E. Real, A. Aggarwal, Y. Huang, and Q. Le, "Aging evolution for image classifier architecture search," in *The Thirty-Third AAAI Conference on Artificial Intelligence*, 2019.

[6] C. Chen, C. Yuan, and C. Chen, "Solving m-modes using heuristic search." in *International Joint Conferences on Artificial Intelligence*, 2016.

[7] A. Kirillov, D. Shlezinger, D. P. Vetrov, C. Rother, and B. Savchynskyy, "M-best-diverse labelings for submodular energies and beyond," in *Advances in Neural Information Processing Systems*, 2015.

[8] C. Chen, V. Kolmogorov, Y. Zhu, D. Metaxas, and C. Lampert, "Computing the m most probable modes of a graphical model," in *Artificial Intelligence and Statistics*, 2013.

[9] H. Pham, M. Y. Guan, B. Zoph, Q. V. Le, and J. Dean, "Efficient neural architecture search via parameters sharing," in *International Conference on Machine Learning*, 2018.

[10] H. Cai, J. Yang, W. Zhang, S. Han, and Y. Yu, "Path-level network transformation for efficient architecture search," in *International Conference on Machine Learning*, 2018.

[11] E. Real, S. Moore, A. Selle, S. Saxena, Y. L. Suematsu, Q. Le, and A. Kurakin, "Large-scale evolution of image classifiers," in *International Conference on Machine Learning*, 2017.

[12] A. J. Piergiovanni, A. Angelova, A. Toshev, and M. S. Ryoo, "Evolving space-time neural architectures for videos," *arXiv preprint abs/1811.10636*, 2018.

[13] K. Kandasamy, W. Neiswanger, J. Schneider, B. Poczos, and E. Xing, "Neural architecture search with bayesian optimisation and optimal transport," in *Advances in Neural Information Processing Systems*, 2018.

[14] H. Liu, K. Simonyan, and Y. Yang, "Darts: Differentiable architecture search," in *International Conference on Learning Representations*, 2019.

[15] J. T. Vogelstein, J. M. Conroy, V. Lyzinski, L. J. Podrazik, S. G. Kratzer, E. T. Harley, D. E. Fishkind, R. J. Vogelstein, and C. E. Priebe, "Fast approximate quadratic programming for large (brain) graph matching," *arXiv preprint arXiv:1112.5507*, 2014.

[16] R. Singh, J. Xu, and B. Berger, "Global alignment of multiple protein interaction networks with application to functional orthology detection," *Proceedings of the National Academy of Sciences*, 2008.

[17] C.-S. Liao, K. Lu, M. Baym, R. Singh, and B. Berger, "Isorankn: spectral methods for global alignment of multiple protein networks," *Bioinformatics*, vol. 25, no. 12, pp. i253–i258, 2009.

[18] G. Kollias, S. Mohammadi, and A. Grama, "Network similarity decomposition (nsd): A fast and scalable approach to network alignment," *IEEE Transactions on Knowledge and Data Engineering*, vol. 24, no. 12, pp. 2232–2243, 2012.

[19] V. Gligorijević, N. Malod-Dognin, and N. Pržulj, "Fuse: multiple network alignment via data fusion," *Bioinformatics*, vol. 32, no. 8, pp. 1195–1203, 2015.

[20] M. Bayati, M. Gerritsen, D. F. Gleich, A. Saberi, and Y. Wang, "Algorithms for large, sparse network alignment problems," in *IEEE International Conference on Data Mining*, 2009.

[21] D. Koutra, H. Tong, and D. Lubensky, "Big-align: Fast bipartite graph alignment," in *IEEE International Conference on Data Mining*, 2013.

[22] E. Malmi, S. Chawla, and A. Gionis, "Lagrangian relaxations for multiple network alignment," *Data Mining and Knowledge Discovery*, vol. 31, no. 5, pp. 1331–1358, 2017.

[23] Z. Si and H. Tong, "Final: Fast attributed network alignment," in *Acm Sigkdd International Conference*, 2016.

[24] M. Heimann, H. Shen, and D. Koutra, "Node representation learning for multiple networks: The case of graph alignment," *arXiv preprint arXiv:1802.06257*, 2018.

[25] P. Drineas and M. W. Mahoney, "Approximating a gram matrix for improved kernel-based learning," in *Conference on Learning Theory*, 2005.

[26] Z.-H. Zhou, *Ensemble methods: foundations and algorithms*. Chapman and Hall/CRC, 2012.

[27] R. Luo, F. Tian, T. Qin, E. Chen, and T.-Y. Liu, "Neural architecture optimization," in *Advances in neural information processing systems*, 2018.

[28] X. Zhang, Z. Huang, and N. Wang, "You only search once: Single shot neural architecture search via direct sparse optimization," *arXiv preprint arXiv:1811.01567*, 2018.

[29] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *arXiv preprint arXiv:1704.04861*, 2017.

[30] X. Zhang, X. Zhou, M. Lin, and J. Sun, "Shufflenet: An extremely efficient convolutional neural network for mobile devices," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2018.