Subarchitecture Ensemble Pruning in Neural Architecture Search

Yijun Bian, Qingquan Song, Mengnan Du, Jun Yao, Huanhuan Chen, *Senior Member, IEEE*, and Xia Hu

***Abstract*— Neural architecture search (NAS) is gaining more and more attention in recent years because of its flexibility and remarkable capability to reduce the burden of neural network design. To achieve better performance, however, the searching process usually costs massive computations that might not be affordable for researchers and practitioners. Although recent attempts have employed ensemble learning methods to mitigate the enormous computational cost, however, they neglect a key property of ensemble methods, namely diversity, which leads to collecting more similar subarchitectures with potential redundancy in the final design. To tackle this problem, we propose a pruning method for NAS ensembles called "subarchitecture ensemble pruning in neural architecture search (SAEP)." It targets to leverage diversity and to achieve subensemble architectures at a smaller size with comparable performance to ensemble architectures that are not pruned. Three possible solutions are proposed to decide which subarchitectures to prune during the searching process. Experimental results exhibit the effectiveness of the proposed method by largely reducing the number of subarchitectures without degrading the performance.**

***Index Terms*— Diversity, ensemble learning, ensemble pruning, neural architecture search (NAS).**

I. Introduction

Designing neural network architectures usually requires manual, laborious architectural engineering, extensive expertise, and high costs. Neural architecture search (NAS), which aims to mitigate these challenges, is attracting increasing attention recently [1]–[3]. However, NAS methods usually require a huge computational effort to achieve an architecture with the expected performance, which is too expensive for many infrastructures and too costly for researchers [4]. Recent work [5]–[7] proposes to employ ensemble methods to mitigate this shortcoming by combining weak subarchitectures trained with lower computational cost into powerful neural architectures. AdaNet, as a prominent example of them, contributes to present a theoretical analysis of the problem of learning both the network architecture and its parameters simultaneously, and proposes the first generalization bounds for the problem of structural learning of neural networks [5], [8].

However, all of them overlook a crucial principle in ensemble methods (i.e., model diversity) in the search for new subarchitectures, which is usually beneficial for creating better model ensembles [9]–[11]. Besides, lots of ensemble pruning methods exploit the diversity property to obtain subensembles with a smaller size than the original ensembles [12], [13]. It has been proven that a few diverse individual learners could even construct a more powerful ensemble learner than the unpruned ensembles [14], [15]. This motivates us to investigate the NAS ensemble pruning problem, where different subensemble architectures are aligned to a smaller but effective ensemble model. Moreover, it is quite challenging to describe the characteristics of diversity for different subarchitectures and decide which one of them should be pruned or kept in the ensemble architecture. First, there are plenty of definitions or measurements for diversity in the ensemble learning community [15]. Unlike the model accuracy, however, there is no well-accepted formal definition of diversity [16]. Second, diversity among individual learners usually decreases as those individual learners approach higher levels of accuracy [17]. Combining some diverse individual learners with some relatively weak ones is usually better than combining accurate ones only because diversity is more important than pure accuracy. Third, selecting the best combination of subarchitectures from an ensemble architecture is NP-complete hard with exponential computational complexity [18], [19]. Thus, how to manage the trade-off between accuracy and diversity properly, and how to select the best subset of ensemble architectures, is a significant problem in the NAS ensemble pruning problems.

Motivated by the characteristic of diversity in ensemble learning, we strive for diverse subensemble architectures at a smaller size, meanwhile, maintaining comparable accuracy performance to the original ensemble architecture without pruning. The idea is to prune the ensemble architecture on-the-fly based on various criteria and keep more valuable subarchitectures in the searching process. Our NAS ensemble pruning method is named as "subarchitecture ensemble pruning in neural architecture search (SAEP)," motivated by AdaNet [5] and ensemble pruning methods, with three proposed criteria to decide which subarchitectures to prune. Note that SAEP also has some differences from typical ensemble pruning problems, because most pruning methods are usually handled on the original ensemble that has been trained, but pruning in SAEP is done in the process of searching rather than after the searching. Moreover, SAEP might lead to distinct deeper architectures than the original one if the degree of diversity is insufficient, which could be a bonus because of pruning. Our contribution in this brief is threefold.

1) We propose a NAS ensemble pruning method to search subensemble architectures at a smaller size that benefits from an essential characteristic, i.e., diversity in ensemble learning. It could achieve comparable accuracy performance to ensemble architectures that are not pruned.
2) Moreover, our proposed method would lead to distinct deeper architectures than the original ensemble architecture that is not pruned if the diversity is insufficient.
3) Experimental results exhibit the effectiveness of the proposed method in largely reducing the number of subarchitectures in ensemble architectures and increasing the diversity while maintaining the final performance.

II. Problem Statement

Notations: In this brief, we denote tensors with bold italic lowercase letters (e.g., $\boldsymbol{x}$), vectors with bold lowercase letters (e.g., $\mathbf{x}$), and

Manuscript received 5 December 2019; revised 17 May 2020 and 12 February 2021; accepted 18 May 2021. Date of publication 18 June 2021; date of current version 1 December 2022. This work was supported in part by the National Key Research and Development Program of China under Grant 2016YFB1000905, in part by the National Natural Science Foundation of China under Grant 91746209, and in part by the Fundamental Research Funds for the Central Universities. *(Corresponding author: Huanhuan Chen.)*

Yijun Bian and Huanhuan Chen are with the School of Computer Science and Technology, University of Science and Technology of China, Hefei 230027, China (e-mail: yjbian@mail.ustc.edu.cn; hchen@ustc.edu.cn).

Qingquan Song, Mengnan Du, and Xia Hu are with the Department of Computer Science and Engineering, Texas A&M University, College Station, TX 77840 USA (e-mail: song_3134@tamu.edu; dumengnan@tamu.edu; hu@cse.tamu.edu).

Jun Yao is with the Data Science and Analytics Department, WeBank, Shenzhen 518000, China (e-mail: junyao@webank.com).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TNNLS.2021.3085299.

Digital Object Identifier 10.1109/TNNLS.2021.3085299

TABLE I
USED SYMBOLS AND DEFINITIONS IN THIS BRIEF

Notation	Definition
$[n]$	the representation of $\{1, \ldots, n\}$ for clarity
$\boldsymbol{x} \in \mathcal{X}$	the input of neural networks
$f(\cdot) \in \mathcal{F}$	the function of a neural network with l layers
n_s	the number of units in the s^{th} layer
$h_{k,j}(\cdot)$	the function of a unit in the k^{th} layer ($k \in [l]$)
$\mathbf{u}_s \in \mathbb{R}^{n_s}$	the weight of the s^{th} layer for the units of the kth layer
$\mathbf{h}_k(\cdot)$	the function vector of units in the kth layer
$\mathbf{w}_k \in \mathbb{R}^{n_k}$	the weight of the k^{th} layer for $f(\cdot)$
$\|\mathbf{w}_k\|_p$	the l_p-norm of $\mathbf{w}_k$ where $p \geqslant 1$
$T \geqslant 1$	the number of iterations in the neural architecture searching process
Γ	a specific complexity constraint based on the Rademacher complexity

scalars with italic lowercase letters (e.g., x). We use $\mathbf{x}^{\mathsf{T}}$ to represent the transpose of a vector. Data/hypothesis spaces are denoted by bold script uppercase letters (e.g., $\mathcal{X}$). We use $\mathbb{R}, \mathbb{P}, \mathbb{E}$ and $\mathbb{I}$ to denote the real space, the probability measure, the expectation of a random variable, and the indicator function, respectively.

We summarize the notations and their definitions in Table I. We follow the notations and the definition of the search space in AdaNet to formulate the problem and introduce the proposed method, as it is one of the most popular ensemble search methods in the NAS literature. It is worth mentioning that the proposed pruning criteria could also be generalized to other ensemble methods, which could be interesting for future research. Let f be a neural network with l layers searched using AdaNet [5], [8], where each layer would be connected to the previous layers. The output for each $\boldsymbol{x} \in \mathcal{X}$ would connect to all intermediate units, that is,

$$f(\boldsymbol{x}) = \sum_{1 \leqslant k \leqslant l} \mathbf{w}_k \cdot \mathbf{h}_k(\boldsymbol{x}) \tag{1}$$

where $\sum_{k=1}^{l} \|\mathbf{w}_k\|_1 = 1$ and $\mathbf{h}_k = [h_{k,1}, \ldots, h_{k,n_k}]^{\mathsf{T}}$. $h_{k,j}$ is the function of a unit in the kth layer, that is,

$$h_{k,j}(\boldsymbol{x}) = \sum_{0 \leqslant s \leqslant k-1} \mathbf{u}_s \cdot \phi_s\big(\mathbf{h}_s(\boldsymbol{x})\big), \quad k \in [l] \tag{2}$$

where $h_0(\boldsymbol{x}) = \boldsymbol{x}$ is the 0th layer denoted by the input. Note that $\phi_s(\mathbf{h}_s)$ denotes that $\phi_s(\mathbf{h}_s) = \big(\phi_s(h_{s,1}), \ldots, \phi_s(h_{s,n_s})\big)$ where the ϕ_s is assumed to be 1-Lipschitz activation functions, such as the ReLU[1] or sigmoid[2] function [5]. If $\mathbf{u}_s = 0$ for $s < k-1$ and $\mathbf{w}_k = 0$ for $k < l$, this architecture of f will coincide with the standard multilayer feed-forward ones [5].

To investigate the search space $\mathcal{F}$, $\mathcal{H}_k$ is used to denote the family of the function in the kth layer. Let $\widetilde{\mathcal{H}}_k \stackrel{\text{def}}{=} \mathcal{H}_k \cup (-\mathcal{H}_k)$ denote the union of $\mathcal{H}_k$ and its reflection, and let $\mathcal{H} \stackrel{\text{def}}{=} \cup_{k=1}^{l} \widetilde{\mathcal{H}}_k$ denote the union of the families $\widetilde{\mathcal{H}}_k$. Then $\mathcal{F}$ coincides with the convex hull of $\mathcal{H}$, which means that generalization bounds for ensemble methods could be utilized to analyze learning with $\mathcal{F}$ [5]. Therefore, Cortes *et al.* [5], [8] attempted to propose learning guarantees based on a rademacher complexity analysis [24] to guide their design of algorithms.

Although AdaNet attempts to train multiple weak subarchitectures with lower computational costs to comprise powerful neural architectures inspired by ensemble methods [5], the crucial characteristic of diversity brings opportunities to achieve subensemble architectures at a smaller size with diverse subarchitectures, yet still with the comparable performance to an original ensemble architecture generated by AdaNet. Based on the above notions, we formally define the NAS ensemble pruning problem.

Problem Definition 1 (NAS Ensemble Pruning): Given an ensemble architecture $f(\boldsymbol{x}) = \sum_{1 \leqslant k \leqslant l} \mathbf{w}_k \cdot \mathbf{h}_k(\boldsymbol{x}) \in \mathcal{F}$ searched by ensemble NAS methods such as AdaNet, and a training set $S = \{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_m, y_m)\}$ with the size of m, assuming that all training instances are drawn independently and identically distributed (i.i.d.) from a distribution $\mathcal{D}$ over $\mathcal{X} \times \{c_1, \ldots, c_{n_c}\}$ with n_c as the number of labels, the goal is to prune the ensemble architecture f and search for a subensemble architecture of a smaller size, while maintaining comparable performance to the original ensemble architecture f.

III. SUBARCHITECTURE ENSEMBLE PRUNING IN NEURAL ARCHITECTURE SEARCH

In this section, we elaborate on the proposed NAS ensemble pruning method to obtain smaller yet effective neural ensemble architectures. Before pruning the less valuable subarchitectures, we need to generate subarchitectures first. We take advantage of AdaNet [5] here because of its popularity and superiority in ensemble NAS research, and utilize its objective function to generate candidate subarchitectures in the searching process. The objective function for generating new candidates in AdaNet is defined as

$$\mathcal{L}_g(\mathbf{w}) = \hat{R}_{S,\rho}(f) + \Gamma \tag{3}$$

where $\hat{R}_{S,\rho}(f)$ denotes the empirical margin error of function f on the training set S, and Γ denotes a specific complexity constraint.

As the learning guarantee in [5] applies to binary classification, we introduce an auxiliary function $g(\boldsymbol{x}, y, f)$ in (4) to extend the objective to multiclass classification problems consistent with our problem statement, that is,

$$g(\boldsymbol{x}, y, f) = 2\mathbb{I}\big(f(\boldsymbol{x}) = y\big) - 1. \tag{4}$$

In this case, the empirical margin error $\hat{R}_{S,\rho}(f)$ would be

$$\hat{R}_{S,\rho}(f) = \frac{1}{m} \sum_{1 \leqslant i \leqslant m} \mathbb{I}\big(g(\boldsymbol{x}_i, y_i, f) \leqslant \rho\big). \tag{5}$$

Guided by (3), AdaNet only generates new candidates by minimizing the empirical error and architecture complexity, while overlooking the diversity and differences among different subarchitectures. To achieve smaller yet effective ensembles by taking the diversity property into account, we need first to measure the diversity of different subarchitectures so that a corresponding objective function could be derived to guide us for the selection of more valuable subarchitectures during the searching process.

Specifically, we propose three different ways to enhance the diversity of different subarchitectures. Except for the first solution, the latter two provide specific objective quantification where diversity is involved as guidance among different subarchitectures for NAS. Besides, the diversity of subensemble architectures generated by them could be quantified to verify whether these ways work or not.

Our final NAS ensemble pruning method, named as "SAEP," is shown in Algorithm 1. The key difference between SAEP and AdaNet is that SAEP prunes the less valuable subarchitectures based on certain criteria during the searching process (lines 10–11 in Algorithm 1), instead of keeping all of them, as shown in Fig. 1. At the tth iteration ($t \in [T]$) in Algorithm 1, let $f^{(t-1)} = \sum_{1 \leqslant k \leqslant l} \mathbf{w}_k \cdot \mathbf{h}_k$ denote the neural network constructed before the start of the tth iteration, with the depth $l^{(t-1)}$ of f. The first target at the tth iteration is to generate new candidates (lines 3–4) and select the better one to be added in the model of $f^{(t-1)}$ (lines 4–9) because we expect the searching process is progressive. The second target at the tth iteration is to prune the less valuable subarchitectures for $f^{(t)}$ and keep beneficial ones to construct the final architecture (lines 10–11).

[1]The rectified Linear function (ReLU function) [20]–[22] is defined as $g(z) = \max\{0, z\}$.

[2]The sigmoid function [23] is defined as $\sigma(z) = (1)/(1 + e^{-z})$.

Algorithm 1 subarchitecture ensemble pruning in neural architecture search (SAEP)

Input: Dataset $S = (\boldsymbol{x}_i, y_i)_{i=1}^m$
Parameter: Number of iterations T
Output: Final function $f^{(T)}$
Initialize $f^{(0)} = \mathbf{0}$, and $l^{(0)} = 1$.
for $t = 1$ **to** T **do**
  $\mathbf{w}', \mathbf{h}' = \text{argmin}_{\mathbf{w},\mathbf{h}} \mathcal{L}_g(f^{(t-1)} + \mathbf{w} \cdot \mathbf{h})$ s.t. $\mathbf{h} \in \mathcal{H}_{l^{(t-1)}}$.
  $\mathbf{w}'', \mathbf{h}'' = \text{argmin}_{\mathbf{w},\mathbf{h}} \mathcal{L}_g(f^{(t-1)} + \mathbf{w} \cdot \mathbf{h})$ s.t. $\mathbf{h} \in \mathcal{H}_{l^{(t-1)}+1}$.
  if $\mathcal{L}_g(f^{(t-1)} + \mathbf{w}' \cdot \mathbf{h}') \leqslant \mathcal{L}_g(f^{(t-1)} + \mathbf{w}'' \cdot \mathbf{h}'')$ **then**
    $f^{(t)} = f^{(t-1)} + \mathbf{w}' \cdot \mathbf{h}'$.
  else
    $f^{(t)} = f^{(t-1)} + \mathbf{w}'' \cdot \mathbf{h}''$.
  end if
  Choose $\mathbf{w}_p$ *based on one certain criterion, i.e., picking randomly in* PRS, $\mathcal{L}_d(\mathbf{w})$ *of (6) in* PAP, *or* $\mathcal{L}_e(\mathbf{w}_i)$ *of (16) in* PIE.
  Set $\mathbf{w}_p$ *to be zero.*
end for

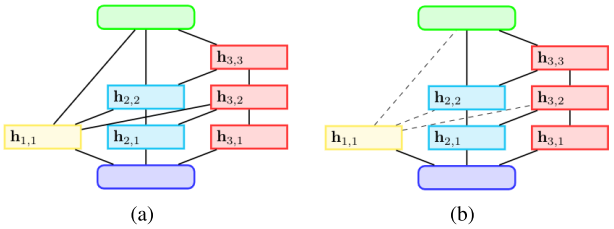

Fig. 1. Difference between SAEP and AdaNet during the incremental construction of neural architectures. Layers in blue and green indicate the input and output layers, respectively. Units in yellow, cyan, and red are added at the first, second, and third iteration, respectively. (a) AdaNet [5]: a line between two blocks of units indicates that these blocks are fully connected. (b) SAEP: only some valuable blocks are kept (those that will be pruned are denoted by black dashed lines), which is the key difference from AdaNet. The criteria used to decide which subarchitectures will be pruned have three proposed solutions in our SAEP, i.e., PRS, PAP, and PIE.

To evaluate the most valuable subarchitectures, we propose three solutions to tackle this problem. Now we introduce them to decide which subarchitectures are less valuable to be pruned.

A. Pruning by Random Selection

The first solution, named as "pruning by random selection (PRS)," is to randomly prune some of the subarchitectures in the searching process, with one difference from other solutions. In PRS, we firstly decide randomly whether or not to pick one of the subarchitectures to be pruned; if we indeed decide to prune one of them, the objective to decide which subarchitectures to prune is random as well, instead of the specific objective in the next two solutions.

However, there is no specific objective for PRS to follow in the pruning process. That might lead to a situation where some valuable subarchitectures are pruned as well. Therefore, we need to find more explicit objectives to guide our pruning.

B. Pruning by Accuracy Performance

To measure different subarchitectures better, we propose the second pruning solution based on their accuracy performance. This method is named as "pruning by accuracy performance (PAP)." To choose the valuable subarchitectures from those individual subarchitectures in the original model, this second optional objective function for this target is defined as

$$\mathcal{L}_d(\mathbf{w}) = \frac{1}{m} \sum_{1 \leqslant i \leqslant m} [g(\boldsymbol{x}_i, y_i, f) - g(\boldsymbol{x}_i, y_i, f - \mathbf{w} \cdot \mathbf{h})] \tag{6}$$

where $\mathbf{h}$ is the subarchitecture corresponding to the weight $\mathbf{w}$. The target is to pick up the $\mathbf{w}$ and $\mathbf{h}$ by minimizing (6) and prune them if their loss is less than zero. The reason why we do this is that the generalization error of gathering all subarchitectures is defined as

$$R(f) = \mathbb{E}_{(\boldsymbol{x},y)\sim\mathcal{D}}\big[\mathbb{I}\big(g(\boldsymbol{x}, y, f) \leqslant 0\big)\big] \tag{7}$$

if the jth subarchitecture is excluded from the final architecture, the generalization error of the pruned subensemble architecture will become

$$R(\bar{f}_j) = \mathbb{E}_{(\boldsymbol{x},y)\sim\mathcal{D}}[\mathbb{I}(g(\boldsymbol{x}, y, f - \mathbf{w}_j \cdot \mathbf{h}_j) \leqslant 0)]. \tag{8}$$

Then, if we expect the pruned architecture works better than the original one, we need to make sure that $R(f) - R(\bar{f}_j) \geqslant 0$, that is,

$$\mathbb{E}_{(\boldsymbol{x},y)\sim\mathcal{D}}\big[g(\boldsymbol{x}, y, f) - g(\boldsymbol{x}, y, f - \mathbf{w}_j \cdot \mathbf{h}_j)\big] \leqslant 0. \tag{9}$$

Therefore, if the jth subarchitecture meeting (9) is excluded from the final architecture, the performance will not be weakened and could be even better than the original one. The hidden meaning behind (9) is that the final architecture makes mistakes; however, the pruned architecture that excludes the jth subarchitecture will work correctly. These subarchitectures that make too serious mistakes to affect the final architecture negatively would be expected to be pruned, leading to our loss function (6). In this case, we could *improve the performance of the final architecture without breaking the learning guarantee*.

However, this objective in (6) only considers the accuracy performance of different subarchitectures and misses out on the crucial characteristic of diversity in ensemble methods. Therefore, we need to find an objective to reflect accuracy and diversity both.

C. Pruning by Information Entropy

To consider accuracy and diversity simultaneously, we propose another strategy, named "pruning by information entropy (PIE)." The objective is based on information entropy. For any subarchitecture $\mathbf{w}_j$ in the ensemble architecture, $\boldsymbol{w}_j = \mathbf{w}_j \cdot [\mathbf{h}_j(\boldsymbol{x}_1), \ldots, \mathbf{h}_j(\boldsymbol{x}_m)]^\mathsf{T}$ represents its classification results on the dataset S. $\mathbf{y} = [y_1, \ldots, y_m]^\mathsf{T}$ is the class label vector. Notice that $H(\cdot)$ and $H(\cdot,\cdot)$ are the entropy function and the joint entropy function, respectively, that is,

$$H(\boldsymbol{w}_i) = -\sum_{w \in \boldsymbol{w}_i} p(w) \log p(w) \tag{10}$$

$$H(\boldsymbol{w}_i, \mathbf{y}) = -\sum_{w \in \boldsymbol{w}_i} \sum_{y \in \mathbf{y}} p(w, y) \log p(w, y). \tag{11}$$

To exhibit the relevance between this subarchitecture and the class label vector, the normalized mutual information [25]

$$\begin{aligned} \text{MI}(\boldsymbol{w}_i, \mathbf{y}) &= \frac{\text{I}(\boldsymbol{w}_i; \mathbf{y})}{\sqrt{\text{H}(\boldsymbol{w}_i)\text{H}(\mathbf{y})}} \\ &= \frac{\sum_{w \in \boldsymbol{w}_i, y \in \mathbf{y}} p(w, y) \log \frac{p(w,y)}{p(w)p(y)}}{\sqrt{\sum_{w \in \boldsymbol{w}_i} p(w) \log p(w) \sum_{y \in \mathbf{y}} p(y) \log p(y)}} \end{aligned} \tag{12}$$

is used to imply its accuracy. Note that

$$\begin{aligned} \text{I}(\boldsymbol{w}_i; \mathbf{y}) &= \text{H}(\boldsymbol{w}_i) - \text{H}(\boldsymbol{w}_i|\mathbf{y}) \\ &= \sum_{w \in \boldsymbol{w}_i, y \in \mathbf{y}} p(w, y) \log \frac{p(w, y)}{p(w)p(y)} \end{aligned} \tag{13}$$

is the mutual information [26]. To reveal the redundancy between two subarchitectures ($\mathbf{w}_i$ and $\mathbf{w}_j$) in the ensemble architecture, the normalized variation of information [25]

$$\begin{aligned} \text{VI}(\boldsymbol{w}_i, \boldsymbol{w}_j) &= 1 - \frac{\text{I}(\boldsymbol{w}_i; \boldsymbol{w}_j)}{\text{H}(\boldsymbol{w}_i, \boldsymbol{w}_j)} \\ &= 1 - \frac{\sum_{w \in \boldsymbol{w}_i, y \in \mathbf{y}} p(w, y) \log \frac{p(w,y)}{p(w)p(y)}}{-\sum_{w \in \boldsymbol{w}_i, y \in \mathbf{y}} p(w, y) \log p(w, y)} \end{aligned} \tag{14}$$

is used to indicate the diversity between them. The objective function for handling the trade-off between diversity and accuracy of two subarchitectures is defined as

$$\mathcal{L}_p(\mathbf{w}_i, \mathbf{w}_j) = (1-\alpha)\mathrm{VI}(\boldsymbol{w}_i, \boldsymbol{w}_j) + \alpha \frac{\mathrm{MI}(\boldsymbol{w}_i, \mathbf{y}) + \mathrm{MI}(\boldsymbol{w}_j, \mathbf{y})}{2} \quad (15)$$

if $\mathbf{w}_i \cdot \mathbf{h}_i \neq \mathbf{w}_j \cdot \mathbf{h}_j$, otherwise $\mathcal{L}_p(\mathbf{w}_i, \mathbf{w}_j) = 0$. Note that α is a regularization factor introduced to balance between these two criteria, indicating their importance as well. Our target is to pick up the $\mathbf{w}$ and $\mathbf{h}$, and prune them by minimizing $\mathcal{L}_e(\mathbf{w})$ in (16), that is,

$$\mathcal{L}_e(\mathbf{w}_i) = \sum_{\mathbf{w}_j \cdot \mathbf{h}_j \in f \setminus \{\mathbf{w}_i \cdot \mathbf{h}_i\}} \mathcal{L}_p(\mathbf{w}_i, \mathbf{w}_j). \quad (16)$$

This loss function considers both diversity and accuracy concurrently according to the essential characteristics in ensemble learning.

IV. Experimental Study

In this section, we describe the experiments to verify the effectiveness of the proposed SAEP method. There are four major questions that we aim to answer: 1) Could SAEP achieve subensemble architectures at a smaller size yet still with comparable accuracy performance to the original ensemble architecture? 2) Could SAEP generate subensemble architectures with more diversity than the original ensemble architecture? 3) What are the impacts of the parameter α on the subensemble architectures generated by PIE? (4) Could PIE generate different subarchitectures from that in the original ensemble architecture?

A. Three Image Classification Datasets

The three image classification datasets that we employ in the experiments are all publicly available. The ImageNet [28] dataset is not included because the cost for it is not affordable for one GPU (NVIDIA GTX 1080) that we use.

1) CIFAR-10 [29]: Sixty thousand 32 × 32 color images in ten classes are used as instances, with 6000 images per class, representing airplanes, automobiles, birds, cats, deer, dogs, frogs, horses, ships, and trucks, respectively. There are 50 000 training images and 10 000 test images.

2) MNIST [30]: Seventy thousand 28 × 28 grayscale images of handwritten digits in ten different classes are used as instances. There are 60 000 instances as a training set and 10 000 instances as a test set. The digits have been size-normalized and centered in a fixed-size image.

3) Fashion-MNIST [31]: Seventy thousand 28 × 28 grayscale images are used as instances, including 60 000 instances for training and 10 000 instances for testing. They are categorized into ten classes, representing T-shirts/tops, trousers, pullovers, dresses, coats, sandals, shirts, sneakers, bags, and ankle boots, respectively.

B. Baseline Methods

To analyze the effectiveness of SAEP, we compare the three proposed solutions (i.e., PRS, PAP, and PIE) with AdaNet [5]. Besides, AdaNet (usually set to use uniform average weights in practice) has a variant to use mixture weights, which we call AdaNet.*W* [33]. Similarly, PRS.*W*, PAP.*W*, and PIE.*W* (i.e., SAEP.*W*) are variants of PRS, PAP, and PIE using mixture weights, respectively. Our baselines include AdaNet and their corresponding variants. Besides, to objectively evaluate the performance of these methods, standard 5-fold cross-validation is used in these experiments, i.e., in each iteration, the entire dataset is split into two parts, with 80% as the training set and 20% as the test set.

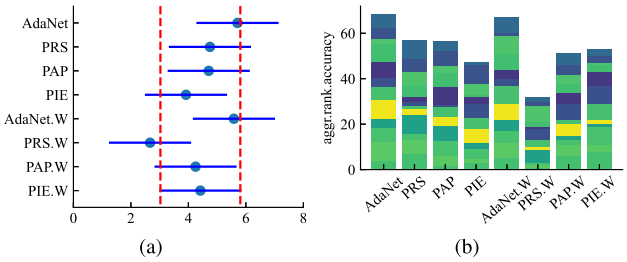

Fig. 2. Comparison of the baseline AdaNet and the proposed SAEP including their variants on the *test accuracy*, using MLPs as subarchitectures for binary classification. (a) Friedman test chart (nonoverlapping means significant difference) [27], which rejects the assumption that "all methods have the same accuracy performance" at the significance level of 10%. (b) Aggregated rank of test accuracy for each method (the smaller the better) [32].

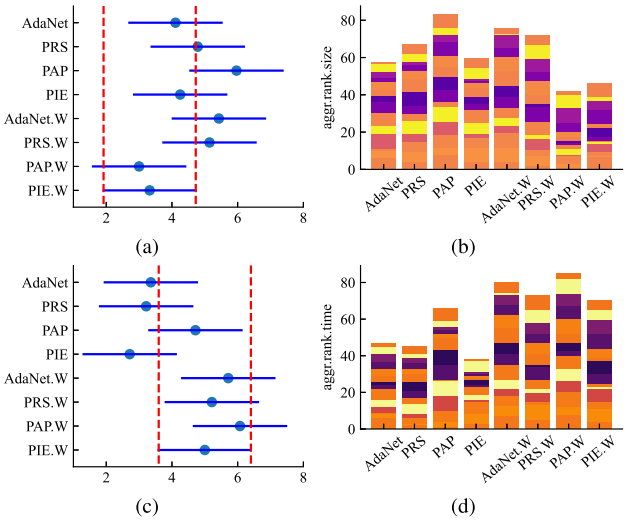

Fig. 3. Comparison of the baseline AdaNet and the proposed SAEP including their variants, using MLPs as subarchitectures for image classification. (a) and (b) Comparison on the *size* of generated (sub-)ensemble architectures. (c) and (d) Comparison on the *time cost* of the searching process. Notice that the Friedman test chart in (a) rejects the assumption that "the size of ensemble architectures of different methods has no significant difference" at 5% significant level; that in (c) rejects the assumption that "the time cost of different methods has no significant difference" at 5% significant level.

C. Experimental Settings

In the same experiment, all methods would use the same kind of subarchitectures in consideration of fairness during the comparisons to verify whether their objectives work well. The optional subarchitectures that we use include multilayer perceptrons (MLPs) and convolutional neural networks (CNNs). Note that those CNNs are only composed of convolution layers with 16 channels, without pooling layers. As for the hyper-parameters in the experiments, the learning rate is set to be 0.003, and cosine decay is applied to the learning rate using a momentum optimizer in the training process. The number of training steps is 5000, and that of the batch size is 64.

We use three datasets mentioned before for image classification. In the multiclass classification scenario, we use all of the categories in the corresponding dataset; in the binary classification scenario, we reduce these datasets by considering several pairs of classes. For example, we consider five pairs of classes in CIFAR-10 (i.e., deer-truck, deer-horse, automobile-truck, cat-dog, and dog-horse), five pairs of classes in Fashion-MNIST (i.e., top-pullover, top-coat, top-shirt, trouser-dress, and sandal-ankle boot), and two pairs of digits in MNIST (i.e., digits 6–9, and 5–8).

TABLE II

EMPIRICAL RESULTS OF ENSEMBLE-ARCHITECTURES' PERFORMANCE FOR BINARY CLASSIFICATION ON CIFAR-10, FASHION-MNIST, AND MNIST DATASETS. EACH METHOD INCLUDES THREE COLUMNS, I.E., THE *Test Accuracy (%)*, THE *Size* OF GENERATED (SUB-)ENSEMBLE ARCHITECTURES, AND THE *Time Cost (Min)* OF THE SEARCHING PROCESS. THE BEST OF THEM ARE INDICATED WITH BOLD FONTS FOR EACH LABEL PAIR (ROW). NOTE THAT SUBARCHITECTURES USED IN THESE EXPERIMENTS ARE MLPS. (a) COMPARISON ON THE *Test Accuracy (%)* PERFORMANCE. (b) COMPARISON ON THE *Size* OF THE (SUB-)ENSEMBLE ARCHITECTURES. (c) COMPARISON ON THE *Time Cost (Min)* OF THE SEARCHING PROCESS.

Label Pair	**Test Accuracy (%)**							
	AdaNet	*PRS*	*PAP*	*PIE*	AdaNet.*W*	*PRS.W*	*PAP.W*	*PIE.W*
digits 6-9	99.85±0.07	99.83±0.05	99.85±0.07	99.85±0.09	99.84±0.05	**99.88±0.04**‡	99.84±0.14†	99.83±0.07
digits 5-8	99.14±0.13	99.18±0.21	99.19±0.18	99.21±0.10‡	99.16±0.18	**99.26±0.15**	99.19±0.18	99.20±0.15
top-pullover	97.03±0.47	97.06±0.15‡	97.03±0.39‡	97.04±0.34‡	97.04±0.30‡	**97.16±0.18**‡	97.08±0.21‡	96.94±0.17
top-coat	98.62±0.14	98.57±0.29†	98.61±0.07	98.64±0.32	98.61±0.22‡	98.61±0.22‡	98.66±0.23	**98.67±0.14**
top-shirt	85.87±0.77	86.36±0.82	86.33±0.77‡	86.18±0.80	86.14±0.54‡	**86.48±0.62**‡	86.27±0.69‡	86.36±0.54‡
trouser-dress	98.35±0.16	**98.42±0.16**	98.39±0.17	98.38±0.28	98.30±0.14	98.39±0.14‡	98.41±0.27	98.32±0.24‡
sandal-ankle boot	98.74±0.19	98.78±0.27	**98.79±0.24**	98.71±0.11	98.78±0.14‡	98.71±0.10	98.69±0.19	98.69±0.31†
deer-truck	87.91±0.38	88.01±0.42	87.87±0.85†	87.99±0.67	87.95±0.35‡	**88.05±0.47**	87.93±0.40	87.91±0.49
deer-horse	75.54±1.45	76.22±1.17‡	76.22±1.07‡	**76.62±0.94**‡	76.10±1.28‡	76.31±1.15‡	76.22±0.41‡	76.25±0.46‡
automobile-truck	72.93±0.24	72.82±0.57†	72.91±0.29†	72.84±0.85†	72.58±0.94†	72.78±0.50†	72.90±0.81†	**72.95±1.10**
cat-dog	61.15±0.69	61.11±0.23	61.02±1.24†	60.67±1.05†	**61.63±0.81**	61.60±0.68‡	61.22±0.47‡	61.53±1.34
dog-horse	78.21±0.30	77.95±1.02†	78.29±0.61	**78.44±0.23**‡	78.20±0.81†	78.41±0.71	78.23±0.98	78.38±0.95
t-test (W/T/L)	—	3/7/2	3/6/3	2/6/4	3/4/5	2/4/6	2/6/4	2/8/2
Average Rank	5.71	4.75	4.71	3.92	5.58	2.67	4.25	4.42

[1] The reported results are the average values of each method and the corresponding standard deviation under 5-fold cross-validation on each dataset.
[2] By two-tailed paired *t*-test at 5% significance level, ‡ and † denote that the performance of AdaNet is inferior to and superior to that of the comparative *SAEP* method with their variants, respectively.
[3] The last two rows show the results of *t*-test and average rank, respectively. The "W/T/L" in *t*-test indicates that AdaNet is superior to, not significantly different from, or inferior to the corresponding comparative *SAEP* methods including their variants. The average rank is calculated according to the Friedman test [27].

(a)

Label Pair	**Number of Sub-Architectures**							
	AdaNet	*PRS*	*PAP*	*PIE*	AdaNet.*W*	*PRS.W*	*PAP.W*	*PIE.W*
digits 6-9	**4.20±1.47**	5.80±0.40	6.00±0.89	5.80±0.98	6.20±0.98	5.20±1.17	5.60±0.49	5.00±1.79†
digits 5-8	6.80±0.40	6.00±0.63‡	6.60±0.49	5.80±0.75	6.00±0.63‡	6.20±1.17	**5.60±0.49**‡	6.00±1.10
top-pullover	5.00±0.63	5.40±0.49	5.80±0.98†	5.20±0.98†	5.00±0.89	3.80±1.17	4.20±0.75	**3.20±0.40**‡
top-coat	5.40±0.80	4.80±0.40‡	4.60±0.80‡	5.40±0.80	5.20±0.75‡	5.40±0.49‡	4.40±0.80	**3.00±0.00**‡
top-shirt	5.60±0.49	5.40±0.80	5.60±0.80	5.20±1.47	5.60±1.02	5.80±0.75†	**4.20±0.98**	4.60±1.62
trouser-dress	4.20±1.47	5.20±0.75	5.20±1.17	4.40±1.36	5.00±1.10	4.00±1.79	**4.00±0.63**‡	4.60±1.62†
sandal-ankle boot	5.20±0.75	5.80±1.17†	5.40±1.02†	5.40±1.36†	6.20±0.75†	5.40±0.49	4.80±0.75‡	**3.40±0.80**‡
deer-truck	4.80±1.17	4.80±1.17	5.00±0.89	4.60±1.02‡	5.20±0.75	5.20±1.33†	4.60±0.80‡	**4.20±0.98**‡
deer-horse	4.00±0.63	4.40±0.80†	5.20±1.17†	**3.40±0.80**	5.00±0.00†	5.00±0.00†	5.00±0.63†	5.20±0.75†
automobile-truck	4.40±1.02	4.20±1.47	4.40±0.80‡	**3.20±1.33**	5.00±1.41†	5.00±1.26†	5.20±0.40	4.80±0.75
cat-dog	4.00±1.10	4.00±1.26	4.00±1.26	4.40±1.50†	**3.40±0.49**‡	5.40±0.80	4.60±1.02	3.60±0.49‡
dog-horse	**4.00±1.10**	5.00±0.89	5.40±1.02	5.00±0.63	4.60±0.80	5.00±0.63†	4.20±0.75	5.40±0.80
t-test (W/T/L)	—	2/8/2	3/7/2	3/8/1	3/6/3	5/6/1	1/7/4	3/4/5
Average Rank	3.96	4.79	6.00	4.00	5.38	5.38	3.25	3.25

(b)

Label Pair	**Time Cost (min)**							
	AdaNet	*PRS*	*PAP*	*PIE*	AdaNet.*W*	*PRS.W*	*PAP.W*	*PIE.W*
digits 6-9	**10.92±1.82**	12.35±0.99	12.91±1.31	13.04±0.76	12.74±1.29	12.24±0.92	13.50±0.28	12.61±1.84†
digits 5-8	13.00±0.46	12.25±0.64‡	13.96±0.34†	**12.16±0.49**‡	13.07±0.46†	13.07±0.83†	13.16±0.62†	13.55±1.18†
top-pullover	11.83±0.46	11.93±0.52†	12.78±1.29†	11.73±1.77	11.84±0.98†	11.22±1.01	12.08±0.58†	**11.02±0.28**‡
top-coat	12.00±0.68	11.21±0.34‡	**11.05±0.70**	11.78±1.27	12.42±0.52	11.90±0.67‡	12.04±0.62	11.09±0.17‡
top-shirt	11.80±1.06	12.06±1.18†	12.63±0.85	**10.89±2.53**	12.55±0.66	12.63±0.74	12.04±0.83	12.26±1.34†
trouser-dress	10.88±1.40	11.98±0.61	12.36±1.02†	11.24±1.74†	12.19±0.66	**10.75±1.72**	11.92±0.71	12.25±1.23
sandal-ankle boot	11.75±0.94	12.09±0.99†	12.55±1.04†	**10.20±2.35**	13.03±0.40	12.34±0.47	12.20±0.77	11.34±0.91‡
deer-truck	15.50±1.81	13.32±2.08	11.72±1.01‡	**11.39±0.81**‡	16.74±1.00	16.11±1.24	16.90±0.90	15.51±1.18
deer-horse	14.66±1.09	12.70±0.98‡	11.94±1.08‡	**9.82±0.58**‡	16.22±0.68	15.99±1.08	16.94±1.04	17.12±1.03†
automobile-truck	15.31±1.33	12.64±2.23	11.35±1.15‡	**10.04±1.31**‡	15.87±1.70†	16.42±1.36†	16.52±0.50	16.97±0.89
cat-dog	24.28±17.37	17.17±1.48‡	23.06±11.87‡	75.08±113.67†	**14.45±1.29**‡	78.41±104.25†	113.55±193.13†	35.34±38.05†
dog-horse	16.79±2.68	23.08±12.98†	77.02±116.93†	**16.07±1.02**‡	71.94±108.17†	119.54±189.82†	17.00±1.81	46.69±39.39†
t-test (W/T/L)	—	4/4/4	5/3/4	2/5/5	4/7/1	4/7/1	3/9/0	6/3/3
Average Rank	3.25	3.50	5.17	2.50	5.58	5.00	6.08	4.92

(c)

D. SAEP Could Achieve Ensemble Architectures With Better Performance of Accuracy

In this section, we verify whether the pruned subensemble architectures could achieve comparable performance with the original ensemble architecture. Experimental results are summarized in Tables II and III contain the average test accuracy (%) of each method and the corresponding standard deviation under fivefold cross-validation on each dataset. For instance, each row (dataset) in Table II compares the classification accuracy using subarchitectures with the same type, indicating results with higher accuracy and lower standard deviation by bold fonts. When comparing one method with AdaNet, the one with higher values of accuracy and lower standard deviation would win; otherwise, the winner would be decided based on the significance of the difference in the accuracy performance between the two methods, which is examined by two-tailed paired *t*-test at 5% significance level to tell if two methods have significantly different results. Specifically, two methods end up with a tie if there is no significant statistical difference between them; otherwise, the one with higher values of accuracy would win. The performance of each method is reported in the last two rows of Table II, compared with AdaNet in terms of the average rank and the number of datasets that AdaNet has won, tied, or lost, respectively. We may notice that SAEP achieved better results than AdaNet in most cases, yet with possible larger time cost in a few cases. Therefore, it could be referred that SAEP could generate ensemble architectures with better performance of accuracy. Fig. 2(a) shows that SAEP (indicated by PRS, PAP, and PIE) achieves the same level of accuracy performance as AdaNet at least, and their variants even exhibit better accuracy performance than AdaNet and AdaNet.*W*. Similar results are presented in Fig. 2(b) and Table III.

E. SAEP Leads to Ensemble Architectures With Smaller Size

In this section, we verify whether the pruned subensemble architectures could generate comparable performance of architectures with smaller size. Experimental results are reported in Tables II and III and Figs. 3 and 4. As we can see in Table II, SAEP achieves ensemble architectures with the smallest size in most cases, although the significant difference between different methods might not be as large as that of accuracy, as shown in Fig. 3(a) and (b). Meanwhile, Table II and Fig. 3(c) and (d) present that variants of AdaNet and SAEP might cost more time than themselves. However, considering that SAEP already achieves the comparable performance with AdaNet and that SAEP generates ensemble architectures with smaller size indeed, we believe that our NAS ensemble pruning method is still

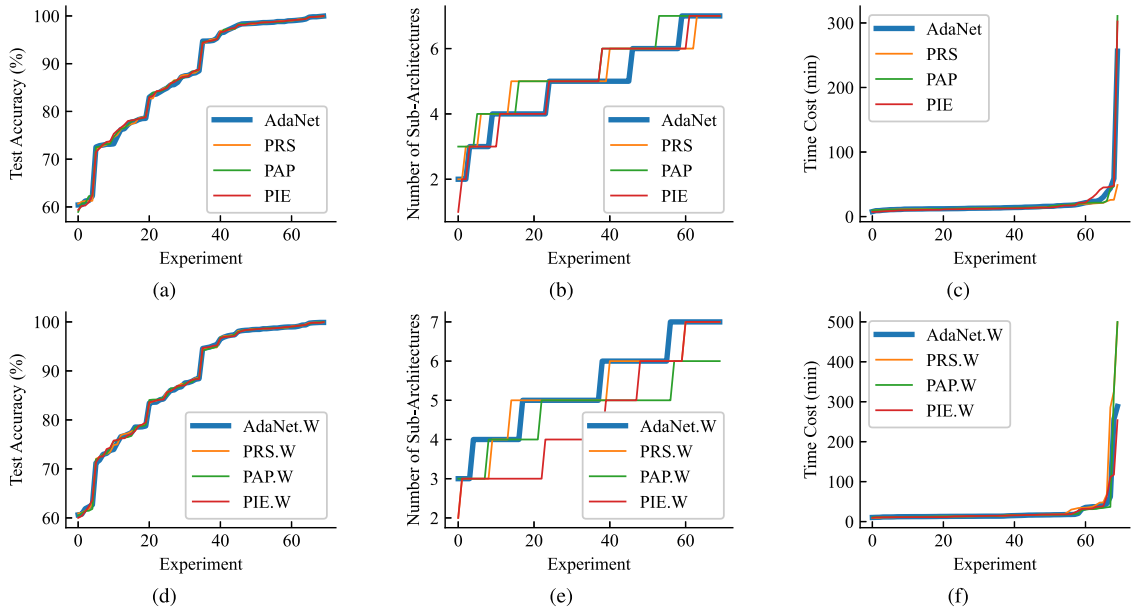

Fig. 4. Comparison of the baseline AdaNet and the proposed SAEP including their corresponding variants, using MLPs as subarchitectures for image classification. The horizontal axis represents empirical results in different specific experiments like different rows in Tables II and III. (a)–(c) Comparison of performance of AdaNet and SAEP. (d)–(f) Comparison of performance of their corresponding variants.

TABLE III

EMPIRICAL RESULTS OF ENSEMBLE-ARCHITECTURES' PERFORMANCE FOR MULTICLASS CLASSIFICATION. EACH METHOD INCLUDES FOUR COLUMNS, I.E., THE *Test Accuracy (%)*, THE *Size* OF GENERATED (SUB-)ENSEMBLE ARCHITECTURES, THE *Diversity* OF THE PRUNED SUBENSEMBLE ARCHITECTURES, AND THE *Time Cost (Min)* OF THE SEARCHING PROCESS. THE BEST OF THEM ARE INDICATED WITH BOLD FONTS FOR EACH DATASET (ROW). NOTE THAT SUBARCHITECTURES USED IN THESE EXPERIMENTS ARE MLPS AND CNNS

Dataset	Test Accuracy (%)							
	AdaNet	*PRS*	*PAP*	*PIE*	AdaNet.*W*	*PRS.W*	*PAP.W*	*PIE.W*
MNIST	94.88±0.22	94.82±0.36†	94.79±0.23†	94.75±0.25†	94.94±0.29	94.66±0.13	94.56±0.28†	**94.94±0.19**‡
Fashion-MNIST	83.74±0.52	83.76±0.88	**83.95±0.50**‡	83.89±0.64	83.81±0.32‡	83.98±0.40‡	**84.24±0.18**‡	83.93±0.21‡
MNIST*	90.54±0.24	90.46±0.25†	90.44±0.15	90.35±0.24†	**90.55±0.18**‡	90.38±0.27†	90.27±0.16	90.23±0.35†
Fashion-MNIST*	81.39±0.43	**81.48±0.30**‡	81.40±0.23‡	81.32±0.45†	81.39±0.26‡	81.41±0.18‡	81.20±0.09	81.05±0.58†
t-test (W/T/L)	—	2/1/1	1/1/2	3/1/0	0/1/3	1/1/2	1/2/1	2/0/2
Average Rank	4.50	3.75	3.75	5.75	3.25	4.00	5.75	5.25

Dataset	Number of Sub-Architectures							
	AdaNet	*PRS*	*PAP*	*PIE*	AdaNet.*W*	*PRS.W*	*PAP.W*	*PIE.W*
MNIST	6.80±0.40	6.60±0.49	7.00±0.00	6.60±0.49	6.60±0.49	6.20±0.98	**5.20±0.75**‡	6.80±0.40
Fashion-MNIST	5.40±1.02	5.60±0.80	5.40±1.02	6.00±0.63	6.00±0.89	5.60±0.80	5.00±0.63‡	**4.00±1.10**
MNIST*	5.80±0.75	4.80±0.75‡	5.60±1.50	4.80±0.40‡	5.40±1.36	5.00±1.41	3.80±1.17	**3.00±0.00**‡
Fashion-MNIST*	5.40±1.36	3.80±1.47‡	6.40±0.49	5.00±0.63‡	5.60±0.49	4.00±0.63‡	4.00±0.89‡	**3.20±0.40**‡
t-test (W/T/L)	—	0/2/2	0/4/0	0/2/2	0/4/0	0/3/1	0/1/3	0/2/2
Average Rank	6.00	3.75	6.63	5.00	6.13	4.00	2.13	2.38

Dataset	Time Cost (min)							
	AdaNet	*PRS*	*PAP*	*PIE*	AdaNet.*W*	*PRS.W*	*PAP.W*	*PIE.W*
MNIST	78.65±89.02	23.42±2.32‡	**21.51±0.16**‡	31.73±12.46‡	81.13±87.66	43.36±12.72‡	92.39±117.81†	33.11±1.94‡
Fashion-MNIST	21.75±1.87	**20.07±0.65**‡	25.50±9.07†	39.30±6.03†	42.92±9.71†	91.41±117.32†	31.77±2.16†	75.92±89.19†
MNIST*	30.73±1.10	29.03±0.74‡	28.04±3.87	**20.47±1.27**‡	30.46±1.72	25.88±2.62‡	29.08±1.23	28.25±0.08‡
Fashion-MNIST*	29.61±1.53	27.74±1.75‡	30.63±1.53†	**20.90±0.83**‡	31.12±0.57†	25.01±3.19	28.22±1.92	28.28±0.19‡
t-test (W/T/L)	—	0/0/4	2/1/1	1/0/3	2/2/0	1/1/2	2/2/0	1/0/3
Average Rank	5.50	2.75	3.50	2.50	7.00	4.25	5.50	5.00

[1] Empirical results with * represent experiments using CNNs as sub-architectures; Empirical results without * represent experiments using MLPs as sub-architectures.
[2] The reported results are the average values of each method and the corresponding standard deviation under 5-fold cross-validation on each dataset.
[3] By two-tailed paired *t*-test at 5% significance level, ‡ and † denote that the performance of AdaNet is inferior to and superior to that of the comparative *SAEP* method with their variants, respectively.
[4] The last two rows show the results of *t*-test and average rank, respectively. The "W/T/L" in *t*-test indicates that AdaNet is superior to, not significantly different from, or inferior to the corresponding comparative *SAEP* methods including their variants. The average rank is calculated according to the Friedman test [27].

meaningful somehow. Similar observations are exhibited in Fig. 4 as well: 1) SAEP could achieve the same level of accuracy performance as AdaNet, as shown in Fig. 4(a) and (d); 2) SAEP could generate ensemble architectures with competitive performance yet smaller size, as shown in Fig. 4(e).

F. PIE Generates Subensemble Architectures With More Diversity

In this section, we verify whether the purpose of increasing the diversity of ensemble architectures is satisfied. We use the normalized information VI in PIE to imply the redundancy between two different subarchitectures, indicating the diversity between them. However, in this experiment, we use another measure named the disagreement measure [34], [35] here to calculate the diversity for the ensemble architecture and the pruned subensemble architectures, because there is no analogous term like VI in PRS and PAP. Note that researchers proposed many other measures to calculate diversity, and the disagreement measure is one of them [36]. We choose the disagreement measure here because this measure is easy to be calculated and understood. The disagreement between two subarchitectures $\mathbf{w}_i$ and $\mathbf{w}_j$ is

$$\text{dis}(\mathbf{w}_i, \mathbf{w}_j) = \frac{1}{m} \sum_{1 \leqslant i \leqslant m} \mathbb{I}(\mathbf{h}_i(\boldsymbol{x}_i) \neq \mathbf{h}_j(\boldsymbol{x}_i)) \tag{17}$$

the diversity of the ensemble architecture f using the disagreement measure is

$$\mathbf{dis}(f) = \frac{2}{l(l-1)} \sum_{\mathbf{w}_i \cdot \mathbf{h}_i \in f} \sum_{\substack{\mathbf{w}_j \cdot \mathbf{h}_j \in f, \\ \mathbf{h}_j \neq \mathbf{h}_i}} \text{dis}(\mathbf{w}_i, \mathbf{w}_j) \tag{18}$$

and the diversity of the subensemble architecture $f \setminus \{\mathbf{w} \cdot \mathbf{h}\}$ could be calculated analogously.

Table IV and Figs. 5 and 6 report their performance with the corresponding disagreement value reflecting the diversity of the whole ensemble architecture as well. Besides, Table IV reports the diversity of the subarchitectures using PIE and other corresponding information. Note that the larger the disagreement is, the larger the diversity of the ensemble architecture or the pruned subarchitecture is. PAP in Table IV achieves better accuracy performance and more diversity concurrently. Similar results are observed in PRS.*W* and PAP.*W* compared with AdaNet.*W* in Table IV, which illustrates that the accuracy of the subensemble architecture could benefit from increasing diversity. Meanwhile, Table IV summarizes that larger subensemble architectures correspond to less diversity sometimes. In addition, Fig. 6 indicate the effect of the α value in (15) on the diversity, the accuracy performance, the time cost, and the size of the subensemble architectures.

TABLE IV

EMPIRICAL RESULTS UNDER DIFFERENT α VALUES ON THE MNIST DATASET FOR BINARY CLASSIFICATION (TO BE SPECIFIC, THE LABEL PAIR OF DIGITS $5-8$), USING MLPS AS SUBARCHITECTURES. EACH METHOD INCLUDES FOUR COLUMNS: THE *Test Accuracy (%)*, THE *Diversity (Disagreement)*, THE *Size* (I.E., THE NUMBER OF SUBARCHITECTURES), AND THE *Time Cost (Min)* OF THE SEARCHING PROCESS. NOTE THAT "ORIG." REPRESENTS ADANET, PRS, PAP, OR PIE; "VARI." REPRESENTS ADANET.*W*, PRS.*W*, PAP.*W*, OR PIE.*W*, CORRESPONDINGLY

	Test Accuracy (%)		Diversity (Disagreement)		Size		Time Cost (min)	
	orig.	vari.	orig.	vari.	orig.	vari.	orig.	vari.
AdaNet	99.86±0.06	99.86±0.05	0.0003±0.0001	0.0005±0.0003	5.60±5.60	5.80±5.80	11.53±0.43	12.37±0.75
PRS	99.83±0.07	99.88±0.05	0.0037±0.0043	0.0019±0.0031	5.40±5.40	4.80±4.80	11.93±0.71	11.16±1.52
PAP	99.87±0.05	99.88±0.04	0.0011±0.0018	0.0003±0.0001	5.80±5.80	5.40±5.40	12.76±1.06	12.89±0.62
PIE ($\alpha = 0.5$)	99.80±0.06	99.85±0.04	0.0037±0.0069	0.0030±0.0053	4.80±4.80	5.80±5.80	12.29±1.14	13.15±0.54
PIE ($\alpha = 0.0$)	99.15±0.23	99.19±0.26	0.0022±0.0004	0.0399±0.0201	7.00±0.00	5.80±0.40	14.53±0.10	13.98±0.63
PIE ($\alpha = 0.05$)	99.16±0.24	99.16±0.13	0.0024±0.0004	0.0280±0.0135	6.60±0.80	6.00±1.10	9.56±2.14	13.60±1.16
PIE ($\alpha = 0.1$)	99.13±0.04	99.21±0.15	0.0019±0.0003	0.0456±0.0250	6.40±0.49	5.40±1.20	13.59±0.80	13.22±0.91
PIE ($\alpha = 0.15$)	99.24±0.14	99.25±0.17	0.0020±0.0004	0.0313±0.0299	6.00±0.89	5.80±0.98	12.95±0.64	14.00±0.75
PIE ($\alpha = 0.2$)	99.14±0.17	99.21±0.06	0.0022±0.0006	0.0574±0.0162	6.20±0.75	6.40±0.49	14.20±0.72	14.58±0.18
PIE ($\alpha = 0.25$)	99.22±0.12	99.15±0.19	0.0020±0.0004	0.0378±0.0294	6.00±0.63	6.40±0.80	13.08±0.61	14.30±0.71
PIE ($\alpha = 0.3$)	99.29±0.12	99.18±0.22	0.0023±0.0003	0.0415±0.0251	7.00±0.00	6.40±0.49	13.16±1.54	14.36±0.41
PIE ($\alpha = 0.35$)	99.19±0.11	99.24±0.09	0.0021±0.0006	0.0210±0.0158	5.60±1.02	5.40±1.20	12.85±0.73	13.20±0.93
PIE ($\alpha = 0.4$)	99.22±0.13	99.20±0.22	0.0022±0.0002	0.0364±0.0288	6.80±0.40	6.20±0.75	11.02±0.12	14.49±0.53
PIE ($\alpha = 0.45$)	99.12±0.12	99.18±0.18	0.0023±0.0009	0.0486±0.0232	6.60±0.80	6.60±0.80	13.62±0.40	14.59±0.44
PIE ($\alpha = 0.55$)	99.17±0.19	99.18±0.20	0.0022±0.0004	0.0454±0.0107	6.00±0.63	6.40±0.49	12.99±0.69	14.45±0.59
PIE ($\alpha = 0.6$)	99.23±0.08	99.19±0.15	0.0022±0.0002	0.0499±0.0275	6.00±1.10	5.60±0.80	9.96±1.46	13.65±1.03
PIE ($\alpha = 0.65$)	99.20±0.14	99.29±0.12	0.0018±0.0003	0.0173±0.0147	6.40±0.80	5.40±1.02	13.14±0.87	13.21±1.08
PIE ($\alpha = 0.7$)	99.23±0.21	99.25±0.12	0.0020±0.0003	0.0234±0.0228	6.60±0.80	6.00±1.26	8.17±0.64	13.78±0.83
PIE ($\alpha = 0.75$)	99.16±0.21	99.22±0.15	0.0022±0.0005	0.0428±0.0216	6.60±0.49	6.00±0.63	13.48±0.51	14.21±0.64
PIE ($\alpha = 0.8$)	99.17±0.15	99.19±0.03	0.0028±0.0017	0.0269±0.0213	6.40±0.49	5.40±1.20	7.80±0.47	13.35±1.05
PIE ($\alpha = 0.85$)	99.20±0.13	99.18±0.17	0.0020±0.0003	0.0411±0.0224	5.60±0.49	6.60±0.49	12.81±0.35	14.45±0.47
PIE ($\alpha = 0.9$)	99.20±0.23	99.28±0.16	0.0017±0.0003	0.0596±0.0172	6.00±0.63	6.00±0.63	7.67±0.92	13.99±0.45
PIE ($\alpha = 0.95$)	99.22±0.19	99.22±0.10	0.0020±0.0003	0.0283±0.0210	5.80±0.40	5.80±1.17	12.51±0.38	13.82±0.80
PIE ($\alpha = 1.0$)	99.19±0.19	99.28±0.15	0.0021±0.0006	0.0483±0.0159	6.20±0.75	6.40±0.49	8.07±0.35	14.34±0.27

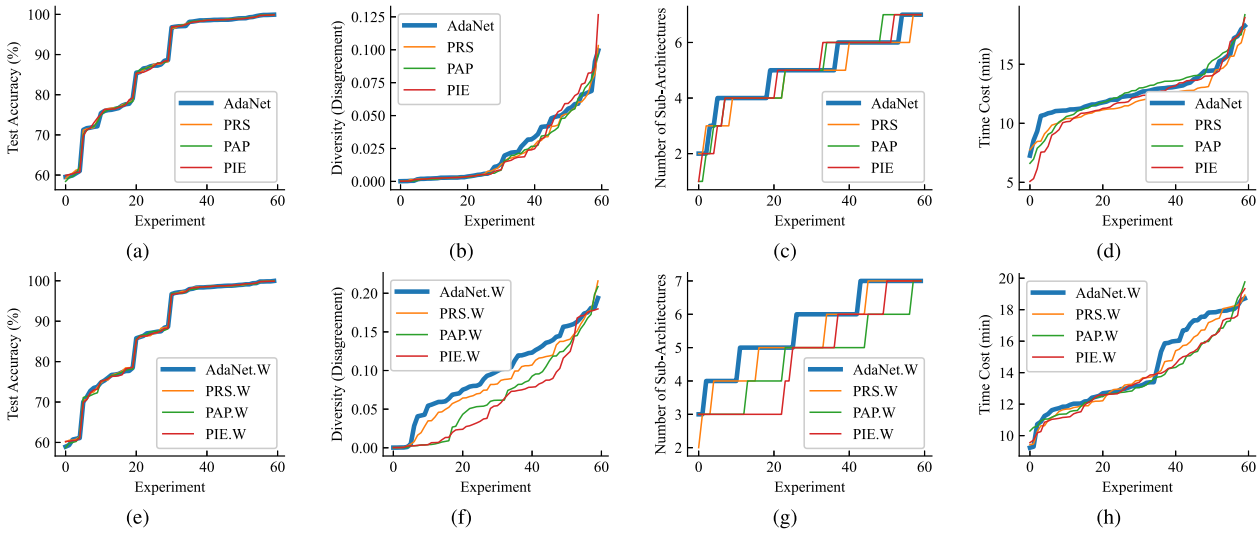

Fig. 5. Comparison of the baseline AdaNet and the proposed SAEP including their corresponding variants, using MLPs as subarchitectures for binary classification. The horizontal axis represents empirical results in different specific experiments like different rows in Tables II and III. (a)–(d) Comparsion of performance of AdaNet and SAEP. (e)–(h) Comparison of performance of their corresponding variants.

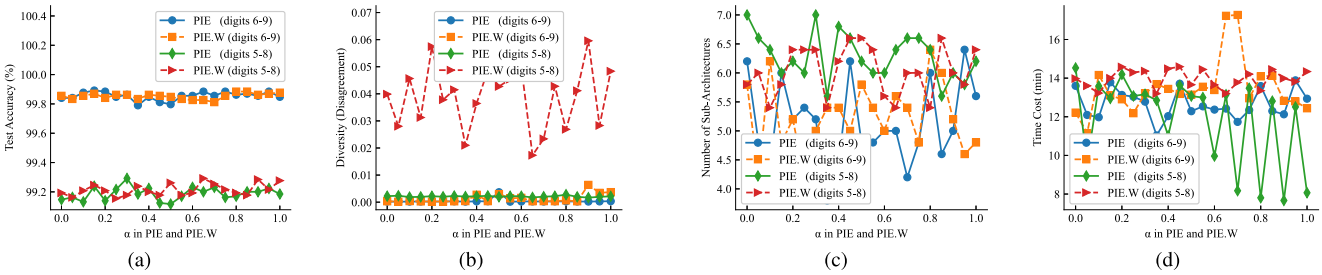

Fig. 6. Effect of different α values in PIE and PIE.*W* for binary classification. (a) Effect of the α value on the test accuracy performance of subensemble architectures. (b) Effect of the α value on the diversity of subensemble architectures, measured by the disagreement measure in (18). (c) Effect of the α value on the size of subensemble architectures. (d) Effect of the α value on the time cost.

G. Effect of the α Value

This section will investigate the effect of the hyper-parameter α in PIE. The value of α indicates the relation between two criteria in (15) as well. To reveal this issue, different α values (from 0.0 to 1.0 with 0.05 steps) are evaluated in the experiments. Fig. 6 exemplify the effect of α on the MNIST dataset, taking the label pairs of digits 5–8 and digits 6–9 as an example. Fig. 6(a) illustrates that the accuracy of subensemble architectures is affected slightly under different α values yet would not cause much accuracy decline. Fig. 6(b) illustrates that the diversity of the subensemble architectures in PIE.*W* is affected under different α values yet without large changes of absolute values; meanwhile, the diversity of that in PIE is

TABLE V

EMPIRICAL RESULTS OF THE EXACT SUBARCHITECTURES THAT ARE KEPT IN THE FINAL SUBENSEMBLE ARCHITECTURE AFTER PRUNING ON THE DIGITS 5–8 LABEL PAIR IN THE MNIST DATASET. EACH METHOD INCLUDES FIVE COLUMNS: THE *Test Accuracy (%)*, THE *Diversity (Disagreement)*, THE *Time Cost (Min)*, THE *Size* (I.E., THE NUMBER OF SUBARCHITECTURES), AND THE *Indexes* OF THE GENERATED SUBARCHITECTURES. NOTE THAT THE SUBARCHITECTURES USED IN THESE EXPERIMENTS ARE MLPS

	Accuracy	Diversity	Time	Size	Indexes
AdaNet	99.86	0.0001	10.71	6	[0,1,3,4,5,6]
PRS	99.92	0.0101	10.54	4	[0,1,2,5]
PAP	99.93	0.0002	13.95	7	[0,1,2,3,4,5,6]
PIE (α=0.25)	99.24	0.0016	13.31	6	[0,1,2,3,4,5]
PIE (α=0.5)	99.89	0.0002	12.89	5	[0,1,2,3,4]
PIE (α=0.75)	99.12	0.0024	13.53	6	[0,1,2,3,5,6]
AdaNet.*W*	99.82	0.0002	11.69	5	[0,1,2,3,6]
PRS.W	99.93	0.0006	13.82	7	[0,1,2,3,4,5,6]
PAP.W	99.86	0.0003	12.47	6	[0,1,3,4,5,6]
PIE.W (α=0.25)	99.20	0.0029	12.91	5	[0,1,2,3,6]
PIE.W (α=0.5)	99.78	0.0004	13.69	6	[0,1,2,3,4,5]
PIE.W (α=0.75)	99.01	0.0445	13.99	6	[0,1,2,3,4,6]

almost not affected under different α values. Fig. 6(c) and (d) presents that the size and time cost of subensemble architectures would be more affected under different α values. Generally, the size and time cost of subensemble architectures in PIE tend to be decreased with the increase of α value.

H. AdaNet Versus SAEP Over the Time Cost

In this section, we compare the time cost of AdaNet and SAEP with their corresponding variants. Experimental results are summarized in Tables II–IV and Figs. 3–6, containing the accuracy on the test set of each method and their corresponding time cost. Although Fig. 3(c) and (d) illustrates that the time cost is not an advantage of SAEP compared with AdaNet while achieving the same level of accuracy, Tables II and III present that SAEP could generate satisfactory subensemble architectures within less time sometimes. Generally, the time cost depends on the number of subarchitectures that are generated during the entire searching process, although the pruning is proceeded through the same process. Therefore, it is quite understandable that SAEP might take a longer time if more subarchitectures are generated during searching. Moreover, Fig. 6(d) presents the effect of different α values in PIE on the time cost of generating subensemble architectures with more diversity.

I. SAEP Could Generate Distinct Deeper Subarchitectures Than AdaNet

In a few cases, we observe that PIE could achieve a larger ensemble architecture than AdaNet, which makes us wonder whether SAEP could lead to distinct architectures from AdaNet. Thus, we dig the subarchitectures that are kept in the final architecture to explore more details deep down inside. As we can see in Table V, the size of subensemble architectures tends to be larger under the lower level of diversity. The reason why PIE (or PIE.*W*) generates distinct deeper subarchitectures might be the diversity is not sufficient for its objective in (16). In this case, the objective would guide the pruning process to search for more distinct deeper subarchitectures to increase diversity.

V. RELATED WORK

In this section, we introduce the NAS briefly. The concept of "NAS" was proposed by Zoph and Le [4] for the very first time. They presented NAS as a gradient-based method to find good architectures. A "controller," denoted by a recurrent network, was used to generate variable-length string which specified the structure and connectivity of a neural network; the generated "child network," specified by the string, was then trained on the real data to obtain accuracy as the reward signal, to generate an architecture with higher probabilities to receive high accuracy [1], [4], [37]. Existing NAS methods could be categorized under three dimensions: search space, search strategy, and performance estimation strategy [2], [38]–[40]. Classical NAS methods yielded chain-structured neural architectures [2], [41], yet ignored some modern designed elements from hand-crafted architectures, such as skip connections from ResNet [42]. Thus, some researchers also attempted to build complex multibranch networks by incorporating those and achieved positive results [43]–[50].

Recently, NAS methods involved ensemble learning are attracting researchers' attention gradually. Cortes *et al.* [5] proposed a data-dependent learning guarantee to guide the choice of additional subnetworks and presented AdaNet to learn neural networks adaptively. They claimed that AdaNet could precisely address some of the issues of wasteful data, time, and resources in NAS since their optimization problem for AdaNet was convex and admitted a unique global solution. Besides, Huang *et al.* [6] specialized subarchitectures by residual blocks and claimed that their BoostResNet boosted over multichannel representations/features, which was different from AdaNet. Macko *et al.* [7] also proposed another attempt named as AdaNAS to utilize ensemble methods to compose a neural network automatically, which was an extension of AdaNet with the difference of using subnetworks comprising stacked NASNet [1], [4] blocks. However, both of them gathered all searched subarchitectures together and missed out on the critical characteristic that ensemble models usually benefit from diverse individual learners.

Moreover, Chang *et al.* [51] proposed Differentiable ARchiTecture Search with Ensemble Gumbel-Softmax (DARTS-EGS) and developed ensemble Gumbel-Softmax to maintain efficiency in searching. Ardywibowo *et al.* [52] constructed an ensemble model to perform the Out-of-Distribution (OoD) detection in their Neural architecture distribution search (NADS), which searched for a distribution of architectures instead of one single best-performing architecture in standard NAS methods. These two methods are not discussed in this brief because they are not assembling subarchitectures during searching.

VI. CONCLUSION

Recent attempts on NAS with ensemble learning methods have achieved prominent results in reducing the search complexity and improving the effectiveness [5]. However, current approaches usually miss out on an essential characteristic of diversity in ensemble learning. To tackle this problem, in this brief, we target the ensemble learning methods in NAS and propose an ensemble pruning method named "subarchitecture ensemble pruning in neural architecture search (SAEP)" to reduce the redundant subarchitectures during the searching process. Three solutions are proposed as the guiding criteria in SAEP that reflect the characteristics of the ensemble architecture (i.e., PRS, PAP, and PIE) to prune the less valuable subarchitectures. Experimental results indicate that SAEP could guide diverse subarchitectures to create subensemble architectures in a smaller size yet still with comparable performance to the ensemble architecture that is not pruned. Besides, PIE might lead to distinct deeper subarchitectures if diversity is insufficient. In the future, we plan to generalize the current method to more diverse ensemble strategies and derive theoretical guarantees to further improve the performance of the NAS ensemble architectures.

REFERENCES

[1] B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le, "Learning transferable architectures for scalable image recognition," in *Proc. CVPR*, Jun. 2018, pp. 8697–8710.

[2] T. Elsken, J. H. Metzen, and F. Hutter, "Correction to: Neural architecture search," in *Automated Machine Learning* (The Springer Series on Challenges in Machine Learning), F. Hutter, L. Kotthoff, and J. Vanschoren, Eds. Cham, Switzerland: Springer, 2019, doi: 10.1007/978-3-030-05318-5_11.

[3] M. Wistuba, A. Rawat, and T. Pedapati, "A survey on neural architecture search," 2019, *arXiv:1905.01392*. [Online]. Available: http://arxiv.org/abs/1905.01392

[4] B. Zoph and Q. V. Le, "Neural architecture search with reinforcement learning," in *Proc. ICLR*, 2017, pp. 1–16.

[5] C. Cortes, X. Gonzalvo, V. Kuznetsov, M. Mohri, and S. Yang, "AdaNet: Adaptive structural learning of artificial neural networks," in *Proc. ICML*, 2017, pp. 874–883.

[6] F. Huang, J. Ash, J. Langford, and R. Schapire, "Learning deep ResNet blocks sequentially using boosting theory," in *Proc. ICML*, 2018, pp. 2058–2067.

[7] V. Macko, C. Weill, H. Mazzawi, and J. Gonzalvo, "Improving neural architecture search image classifiers via ensemble learning," 2019, *arXiv:1903.06236*. [Online]. Available: http://arxiv.org/abs/1903.06236

[8] C. Cortes, M. Mohri, and U. Syed, "Deep boosting," in *Proc. ICML*, 2014, pp. 1179–1187.

[9] H. Chen and X. Yao, "Regularized negative correlation learning for neural network ensembles," *IEEE Trans. Neural Netw.*, vol. 20, no. 12, pp. 1962–1979, Dec. 2009.

[10] H. Chen and X. Yao, "Multiobjective neural network ensembles based on regularized negative correlation learning," *IEEE Trans. Knowl. Data Eng.*, vol. 22, no. 12, pp. 1738–1751, Dec. 2010.

[11] H. Chen and X. Yao, "Evolutionary random neural ensembles based on negative correlation learning," in *Proc. IEEE CEC*, Sep. 2007, pp. 1468–1474.

[12] Y. Bian, Y. Wang, Y. Yao, and H. Chen, "Ensemble pruning based on objection maximization with a general distributed framework," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 31, no. 9, pp. 3766–3774, Sep. 2020.

[13] H. Chen, P. Tino, and X. Yao, "A probabilistic ensemble pruning algorithm," in *Proc. ICDM Workshops*, 2006, pp. 878–882.

[14] H. Chen, P. Tiňo, and X. Yao, "Predictive ensemble pruning by expectation propagation," *IEEE Trans. Knowl. Data Eng.*, vol. 21, no. 7, pp. 999–1013, Jul. 2009.

[15] H. Chen, "Diversity and regularization in neural network ensembles," Ph.D. dissertation, School Comput. Sci., Univ. Birmingham, Birmingham, U.K., 2008.

[16] Y. Bian and H. Chen, "When does diversity help generalization in classification ensembles?" *IEEE Trans. Cybern.*, early access, Feb. 26, 2021, doi: 10.1109/TCYB.2021.3053165.

[17] Z. Lu, X. Wu, X. Zhu, and J. Bongard, "Ensemble pruning via individual contribution ordering," in *Proc. SIGKDD*, 2010, pp. 871–880.

[18] N. Li, Y. Yu, and Z.-H. Zhou, "Diversity regularized ensemble pruning," in *Proc. ECML-PKDD*, 2012, pp. 330–345.

[19] G. Martínez-Muñoz and A. Suárez, "Using boosting to prune bagging ensembles," *Pattern Recognit. Lett.*, vol. 28, no. 1, pp. 156–165, Jan. 2007.

[20] K. Jarrett, K. Kavukcuoglu, M. A. Ranzato, and Y. LeCun, "What is the best multi-stage architecture for object recognition?" in *Proc. ICCV*, Sep. 2009, pp. 2146–2153.

[21] V. Nair and G. Hinton, "Rectified linear units improve restricted Boltzmann machines," in *Proc. ICML*, 2010, pp. 807–814.

[22] X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier neural networks," in *Proc. AISTATS*, 2011, pp. 315–323.

[23] I. Goodfellow, Y. Bengio, A. Courville, and Y. Bengio, *Deep Learning*, vol. 1. Cambridge, MA, USA: MIT Press, 2016.

[24] V. Koltchinskii and D. Panchenko, "Empirical margin distributions and bounding the generalization error of combined classifiers," *Ann. Statist.*, vol. 30, no. 1, pp. 1–50, Feb. 2002.

[25] S. Zadeh, M. Ghadiri, V. Mirrokni, and M. Zadimoghaddam, "Scalable feature selection via distributed diversity maximization," in *Proc. AAAI*, 2017, pp. 2876–2883.

[26] T. Cover and J. Thomas, *Elements of Information Theory*. Hoboken, NJ, USA: Wiley, 2012.

[27] J. Demšar, "Statistical comparisons of classifiers over multiple data sets," *J. Mach. Learn. Res.*, vol. 7, pp. 1–30, Jan. 2006.

[28] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *Proc. CVPR*, Jun. 2009, pp. 248–255.

[29] A. Krizhevsky, "Learning multiple layers of features from tiny images," Dept. Comput. Sci., Univ. Toronto, Toronto, ON, Canada, Tech. Rep. TR-2009, 2009, ch. 3.

[30] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.

[31] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-MNIST: A novel image dataset for benchmarking machine learning algorithms," 2017, *arXiv:1708.07747*. [Online]. Available: http://arxiv.org/abs/1708.07747

[32] C. Qian, Y. Yu, and Z.-H. Zhou, "Pareto ensemble pruning," in *Proc. AAAI*, 2015, pp. 2935–2941.

[33] C. Weill *et al.* (2018). *AdaNet: Fast and Flexible AutoML With Learning Guarantees*. [Online]. Available: https://github.com/tensorflow/adanet

[34] D. B. Skalak, "The sources of increased accuracy for two proposed boosting algorithms," in *Proc. AAAI*, vol. 1129, Aug. 1996, p. 1133.

[35] T. K. Ho, "The random subspace method for constructing decision forests," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 20, no. 8, pp. 832–844, Aug. 1998.

[36] Z.-H. Zhou, *Ensemble Methods: Foundations and Algorithms*. Boca Raton, FL, USA: CRC Press, 2012.

[37] B. Baker, O. Gupta, N. Naik, and R. Raskar, "Designing neural network architectures using reinforcement learning," in *Proc. ICLR*, 2017, pp. 1–18.

[38] K. Kandasamy, W. Neiswanger, J. Schneider, B. Poczos, and E. Xing, "Neural architecture search with Bayesian optimisation and optimal transport," in *Proc. NeurIPS*, Feb. 2018, pp. 2020–2029.

[39] H. Cai, T. Chen, W. Zhang, Y. Yu, and J. Wang, "Efficient architecture search by network transformation," in *Proc. AAAI*, 2018, pp. 1–8.

[40] H. Liu, K. Simonyan, and Y. Yang, "Darts: Differentiable architecture search," in *Proc. ICLR*, 2019, pp. 1–13. [Online]. Available: https://openreview.net/forum?id=S1eYHoC5FX

[41] A. Zela, A. Klein, S. Falkner, and F. Hutter, "Towards automated deep learning: Efficient joint neural architecture and hyperparameter search," in *Proc. ICML Workshop AutoML*, 2018, pp. 1–11.

[42] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. CVPR*, Jun. 2016, pp. 770–778.

[43] H. Cai, J. Yang, W. Zhang, S. Han, and Y. Yu, "Path-level network transformation for efficient architecture search," in *Proc. ICML*, Jun. 2018, pp. 678–687.

[44] E. Real, A. Aggarwal, Y. Huang, and Q. V. Le, "Regularized evolution for image classifier architecture search," in *Proc. AAAI*, 2019, vol. 33, no. 1, pp. 4780–4789.

[45] T. Elsken, J. H. Metzen, and F. Hutter, "Efficient multi-objective neural architecture search via Lamarckian evolution," in *Proc. ICLR*, 2019, pp. 1–23. [Online]. Available: https://openreview.net/forum?id=ByME42AqK7

[46] A. Brock, T. Lim, J. Ritchie, and N. Weston, "Smash: One-shot model architecture search through hypernetworks," in *Proc. NIPS Workshop Meta-Learn.*, 2017, pp. 1–21.

[47] T. Elsken, J. Metzen, and F. Hutter, "Simple and efficient architecture search for convolutional neural networks," in *Proc. NIPS Workshop Meta-Learn.*, 2017, pp. 1–14.

[48] Z. Zhong, J. Yan, W. Wu, J. Shao, and C.-L. Liu, "Practical block-wise neural network architecture generation," in *Proc. CVPR*, Jun. 2018, pp. 2423–2432.

[49] H. Pham, M. Guan, B. Zoph, Q. Le, and J. Dean, "Efficient neural architecture search via parameters sharing," in *Proc. ICML*, 2018, pp. 4095–4104.

[50] Z. Zhong *et al.*, "BlockQNN: Efficient block-wise neural network architecture generation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 43, no. 7, pp. 2314–2328, Jul. 2021.

[51] J. Chang, X. Zhang, Y. Guo, G. Meng, S. Xiang, and C. Pan, "Differentiable architecture search with ensemble gumbel-softmax," 2019, *arXiv:1905.01786*. [Online]. Available: http://arxiv.org/abs/1905.01786

[52] R. Ardywibowo, S. Boluki, X. Gong, Z. Wang, and X. Qian, "NADS: Neural architecture distribution search for uncertainty awareness," in *Proc. ICML*, 2020, pp. 356–366. [Online]. Available: https://openreview.net/forum?id=rJeXDANKwr