Intra-Ensemble in Neural Networks

[1]Yuan Gao*, [2]Zixiang Cai*, [2]Yimin Chen,
[1]Wenke Chen, [3]Kan Yang, [1]Chen Sun, [1]Cong Yao
[1]Megvii Inc (Face++), [2]Beihang University, [3]Peking University
gaoyuan, chenwenke, sunchen@megvii.com, caizixiang@buaa.edu.cn,
minwelldht@gmail.com, kanyang@pku.edu.cn, yaocong2010@gmail.com

Abstract

*Improving model performance is always the key problem in machine learning including deep learning. However, stand-alone neural networks always suffer from marginal effect when stacking more layers. At the same time, ensemble is a useful technique to further enhance model performance. Nevertheless, training several independent stand-alone deep neural networks costs multiple resources. In this work, we propose **Intra-Ensemble**, an end-to-end strategy with stochastic training operations to train several sub-networks simultaneously within one neural network. Additional parameter size is marginal since the majority of parameters are mutually shared. Meanwhile, stochastic training increases the diversity of sub-networks with weight sharing, which significantly enhances intra-ensemble performance. Extensive experiments prove the applicability of intra-ensemble on various kinds of datasets and network architectures. Our models achieve comparable results with the state-of-the-art architectures on CIFAR-10 and CIFAR-100.*

1. Introduction

Ensemble learning has been proved impactful in traditional machine learning [1, 2]. Concurrently, it is widely applied in deep learning as well. Many works [3, 4, 5, 6, 7] utilize ensemble to enhance final performance on different tasks. Nevertheless, most applications of ensemble in deep learning follow the traditional strategy, simply combining several individually trained neural networks. This strategy introduces multiple extra parameters and computational resources, which is extravagant for most practical applications. On the other hand, the design of many famous CNN architectures was partly inspired by the idea of ensemble. The most famous Inception series [8, 9, 10] concatenate different branches with various filter types or depths in each layer. Other architectures like ResNet [11] and DenseNet [12] sum or concatenate different features from previous layers. With such feature-map ensemble skills, even though each branch only contains partial features, the combination of them has a generalized view of all previous outputs. SENet [13] focuses on channel relationship and uses SE block to recalibrate channel-wise features. This work also inspires us to employ different combination skills to enhance ensemble.

*Authors contribute equally.

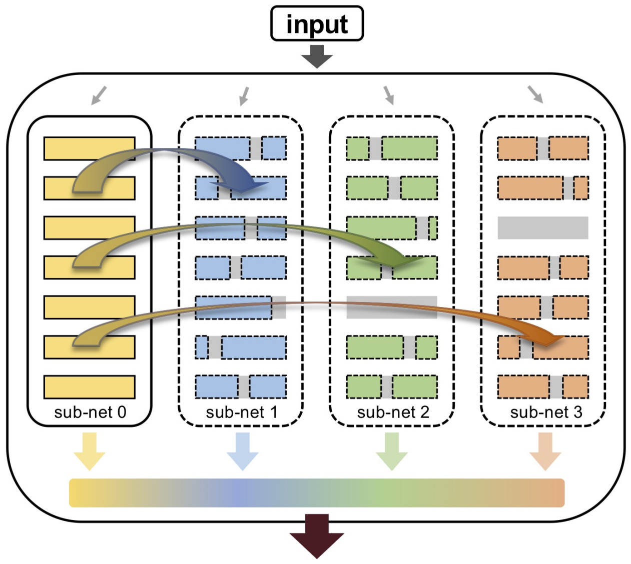


Figure 1: Intra-Ensemble Network (IENet). Sub-net 0 indicates the original network with all channels and layers. Sub-net 1, sub-net 2, sub-net 3 share different channels and layers of the original network.

Our work is mainly inspired by one-shot model [14] and slimmable neural networks [15]. One-shot model proposes to train a neural network with multiple optional operations in each position with certain drop probability. At evaluation time, operations are selected randomly to check which sub-network leads to the best accuracy. Different sub-networks are naturally generated by keeping different operations at each position. The positive correlations make it possible to estimate the stand-alone model's accuracy using its corresponding sub-network in one-shot model. The sampled sub-network with best evaluation performance can be regarded as the best architecture of stand-alone network. However, owing to relatively large search space, there are thousands of underlying combinations to form various sub-networks. Parameters shared in sub-networks conflict with each other, leading to a serious accuracy decrease compared to corresponding stand-alone networks. Recently, Yu et al. [15] find a general method to train a single network executable at different widths by switchable batch normalization(S-BN). This work makes it available to train several weight-sharing sub-networks within one neural network without loss of performance. Different sub-networks mutually share weights so total parameter size is nearly the same as the stand-alone one.

Based on the studies introduced above, we attempt to make use of the redundancy in deep neural networks instead of pruning it. We verify the possibility to train one neural network into several sub-networks with different widths and depths. The sketch-map in figure 1 shows the relation between original network and sub-networks and the conception of intra-ensemble. However, training a switchable network directly will create a series of sub-networks with similar properties, which is obviously harmful to ensemble. For this reason, the key problem is to train high-accuracy and low-correlation sub-networks to keep the generalization ability [16]. Once networks have uncorrelated outputs to solve different kinds of hard cases, ensemble can take effect and contribute to the final performance.

To this end, we propose intra-ensemble with several stochastic training operations called random cut(RC), random offset(RO), shuffle channel(SC) upon width, and random skip(RS), shuffle layer (SF) upon depth to produce diversity for sub-networks. These stochastic training operations successfully diversify the sub-networks. Based on high-accuracy and diversity, we implement simple combination strategies on sub-networks rather than picking up the best one described in one-shot method. We interestingly find that, although sub-networks may suffer a negligible accuracy drop, intra-ensemble result can still surpass the stand-alone neural network with almost same parameter size. Another important issue is time consumption. Giving several sub-networks, training them one-by-one like previous works will greatly increase training time. So we provide independent data loaders for each sub-network. Benefited from weight sharing, this simple modification accelerates training speed significantly.

The main contributions of this work are as follows:

- we verify the feasibility of efficiently training one neural network into several sub-networks with different depths and widths while keeping accuracy.
- we propose intra-ensemble with several stochastic training operations to significantly increase the diversity of sub-networks and boost performance under limited parameter size.
- Through extensive experiments we show the applicability of intra-ensemble on various kinds of datasets and network architectures. Our IENets achieve highly competitive results on CIFAR-10 and CIFAR-100.

2. Related Works

Ensemble learning Ensemble learning [17, 18, 19, 20] is a traditional machine learning technique to break through the performance limitation of a single model. These techniques increase the stability and generalization ability of the final model by reducing their variance, noise, and bias. There are three essential ensemble learning methods. Bootstrap aggregating or so-called bagging [21] trains several models with differently distributed data to reduce primally the global variance. Boosting [22] technique iteratively trains several weak models to reduce the bias of the entire model. Stacking [23] technique trains an extra model based on the results of several weak models to generalize the final result. Ensemble learning can also be applied to deep neural networks [7, 4, 5, 6]. Performance can be further enhanced by combining several well trained neural networks.

Neural architecture search Recently, neural architecture search(NAS) [24, 25, 26] has shown great improvement in designing smaller and more accurate network architectures. Under carefully designed search space and search algorithms, the searched architecture can attain a good trade-off between accuracy and latency. One-shot models [14, 27] train a cumbersome network with different optional operations each layer to approximately estimate the stand-alone model using its corresponding sub-network. This work inspires us to utilize the diversity and flexibility of one-shot model while maintaining the accuracy of its sub-networks. DARTS [28], ProxylessNas [29] and FBNet [30] propose differentiable strategies to implement efficient search of network architecture without any controller or hypernetwork.

Parameter sharing Guided by the thought of minimizing training cost, parameter sharing succeed in improving model performance while reducing model size. This idea has been used in some works on NAS and network morphism. ENAS [31] and EPNAS [32] increase training efficiency by sharing weights and One-shot [14] uses shared weights cumbersome network with optional operations to estimate the performance of stand-alone child networks. Meanwhile, some works apply network morphism to transfer information between networks [33, 34]. Based

Algorithm 1: Training procedure of IENet M

```
1  Definition Define width list $W$ and depth list $D$ for IENet $M$.
2  Initialization Initialize $M$ with independent S-BN parameters for each width $w$ in $W$ and the combination layer $V$.
3  for $i \leftarrow 1$ to $n_{iters}$ do
4      Initialize an empty list $L_{ensemble}$ as the recorder of sub-network logits.
5      for width $w_i$ in $W$ do
6          Switch to the BN adapting current width $w_i$.
7          for depth $d_j$ in $D$ do
8              Arrange the layers for sub-network $M_{ij}$ according to $d_j$.
9              Execute sub-network $\hat{y}_{ij} = M_{ij}(x)$ and append $\hat{y}_{ij}$ to $L_{ensemble}$.
10             Compute loss, $loss_{ij} = criterion(\hat{y}_{ij}, y)$.
11             Compute gradients, $loss_{ij}.backward()$.
12         end
13     end
14     Execute combination layer, $\hat{y}_v = V(L_{ensemble})$.
15     Compute loss and gradient for $V$, $loss_v = criterion(\hat{y}_v, y)$; $loss_v.backward()$.
16     Update weights, $optimizer.step()$; $voting\_optimizer.step()$.
17 end
```

on network morphism, Elsken et al. [4] search for well-performing CNN architectures based on a simple hill climbing procedure. Another inspiring work called slimmable neural networks [15] shows that it is feasible to train several sub-networks within a single network without great loss of accuracy. In our work, we make use of its switchable techniques to train weight sharing sub-networks.

3. Train Sub-networks within One Network

Parameter redundancy can be tactfully utilized by training several sub-networks within one neural network while sharing most weights. Here we define W, a list of width ratios and D, a list of convertible depths passing or skipping different layers. Our work presents an end-to-end strategy to train a single neural network M into a set of sub-networks $M_S = \{M_{wd}\}_{w \in W, d \in D}$ with different widths and depths, as shown in figure 2. Algorithm 1 illustrates a detailed implementation of this strategy.

Width For any width ratio $w_i \in W$, w_i is the ratio of used channels per layer. To be specific, giving a layer with n channels, if $w_i = 0.8$, the corresponding *i-th* sub-network use $0.8n$ channels of the original network in this layer. Still, naive training or incremental training(a.k.a. progressive training) [35] with different widths will cause a great performance decline for each sub-network. To address this issue, we use slimmable [15] as a reference and borrow their idea of switchable batch normalization(S-BN). With S-BN, we can easily train one neural network at different widths with a marginal increase of parameter size, while keeping their high performance.

Depth As for depth, inspired by incremental training method [36], we assign different combinations of layers

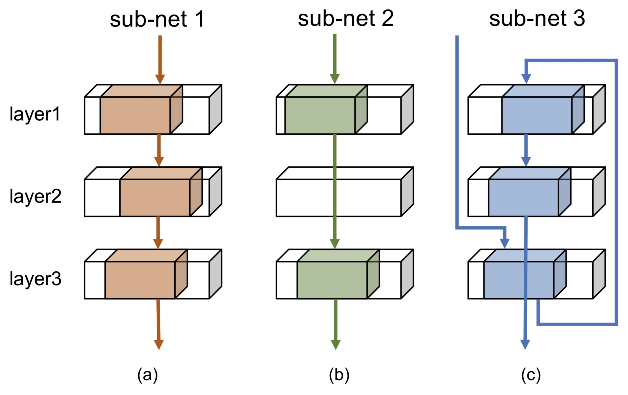


Figure 2: Different sub-networks in one neural network. Arrows indicate data flow.

to different sub-networks. Given a depth list D, for any $d_i \in D$, d_i is a list of layer indexes indicating layers used in the forward and backward propagation in its corresponding sub-network. For example, if we design to train a 5 layer network M into 3 sub-networks $M_S = \{M_1, M_2, M_3\}$ with different depths, the depths list may be $d_1 = \{l_1, l_2, l_3, l_4, l_5\}$, $d_2 = \{l_1, l_2, l_3, l_5\}$, $d_3 = \{l_1, l_2, l_4, l_5\}$, corresponding to M_1, M_2, M_3.

Although sub-networks are trained with different widths and depths, the vast majority of network parameters are mutually shared. Layers with different widths employ independent batch normalization to accumulate their own feature statistics, which ensures stability in both training and inference. With several sub-networks available within one neural network, intra-ensemble can be applied to further enhance

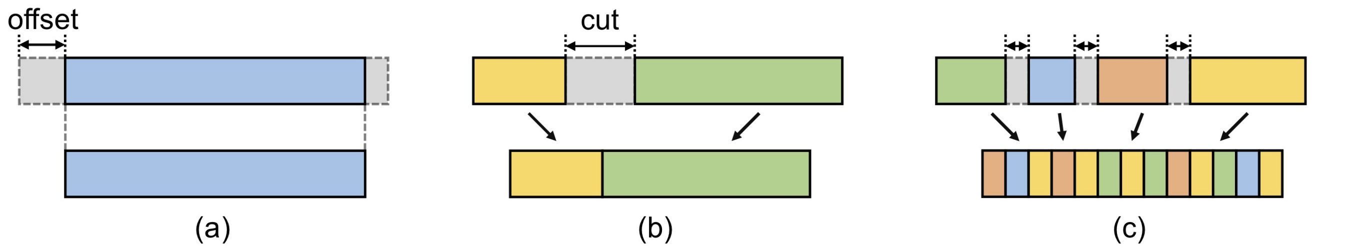


Figure 3: Different stochastic training operations increase the diversity of sub-networks. (a) random offset. (b) random cut. (c) shuffle channel.

network performance.

Accelerate training with shuffled data Previous works train the whole network with same data for each sub-network per batch. Such naive training is inefficient and wasteful because sub-networks share most weights with each other. Shuffling data is a special trick to deal with the issue above. We use different data loaders for different sub-networks. Each data loader is shuffled differently to ensure different permutations. With shuffled data, sub-networks with shared weights are able to learn more information in each batch.

As shown in figure 4, sub-networks training with shuffled data converge faster from the start of training process. According to our experiments, while switching from shuffled data to normal data, the training accuracy will meet a slight drop due to the change of feature statistics. This accuracy drop is more ignorable in the early epochs of training. Moreover, the acceleration effect of shuffling data is more noticeable during the early stages of training. Thus, in order to ensure the final performance, we limit the usage of shuffling data in the early epochs, usually one tenth of the total epochs, according to our experiments.

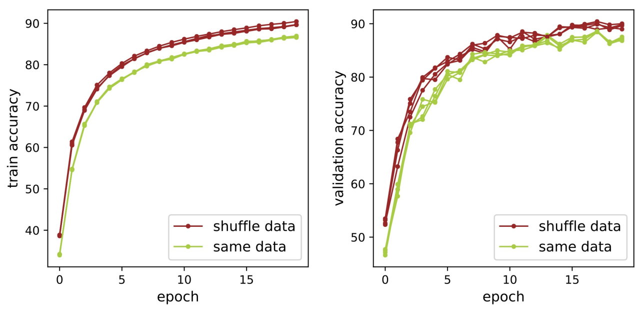


Figure 4: Shuffling data speeds up training process.

4. Intra-Ensemble in One Neural Network

Naively trained sub-networks usually converge towards similar results because they share a large part of parameters with each other. Meanwhile, ensemble demands a set of diverse model to go into effect. So the crucial point is to increase the diversity among different sub-networks. In this section, we introduce intra-ensemble, a training strategy with several stochastic training operations to significantly enhance ensemble performance within one neural network.

4.1. Stochastic Training for Sub-networks

In order to reduce the homogeneity among sub-networks, we resolve the problem into two aspects, width and depth. Different widths provide different filters for sub-networks, while different depths allow sub-networks to pass order-varying blocks or layers.

Our main contributions and experiments focus on the aspect of width. We propose the following three stochastic training operations shown in figure 3 to increase sub-network diversity on width and one operation on depth. Our operations are mainly implemented on channel indexing. Suppose we have a layer l with c channels in total, with a sub-network containing $n(0 < n \leq c)$ filters in this layer. The operations are described as follows:

Random cut(RC) Random cut means randomly cutting out a continuous part of channels from all channels for sub-network. If we cut out p percent channels with a cut index t, we will block out channels with index $[t, t+1, ..., t+pc)$, channels in $[0, ..., t)$ and $[t + pc, c)$ are remained for the forward and backward propagation of corresponding sub-network.

Random offset(RO) Random offset operation sets offsets for sub-network channels, instead of simply choosing channels starting from the head in [15]. If one sub-net layer use p percent of total channels c with offsets t, its channel index list will be $[t, t + 1, t + 2, ..., t + pc)$. The constraint of offset t is $0 \leq t < c - pc$.

Shuffle channel(SC) Shuffle channel is inspired by ShuffleNet [37]. We randomly choose different lists of channel indexes and concatenate the features with shuffled order for sub-network layers. With Shuffle channel operation, different sub-network will use different channels with various order. In this way, diversity can be greatly enhanced.

Depth operation In the aspect of depth, we use the combination of two techniques to diversify the sub-networks, namely random skip(RS) and shuffle layer(SL). With random skip, each sub-network randomly skips one or a few layers of the whole network. This operation provides different depths for sub-networks. Shuffle layer means that in each sub-network, we partly shuffle the order of layers within each stage. In our experiments, when sub-networks have different depths in our model, we always combine RS and SL to ensure diversity.

Similarity Here we define a simple metric called similarity noting $\mathcal{S}$. Given test dataset with N test images, if there are K images with same outputs from all sub-networks, the similarity $\mathcal{S}$ is

$$\mathcal{S} = (N - K)/N$$

The trade-off between sub-networks' performance and similarity $\mathcal{S}$ should be carefully considered. RC reaches the best results according to our evaluation, so we choose **Random Cut** as the most recommended approach and primarily report its performance in different classification tasks.

4.2. Combination Strategy

Since we already have several high-performance and diverse sub-networks, the next step is to decide how to combine them properly. We simply apply the basic combination strategies of ensemble learning as follows:

Consider the C classes softmax [38] outputs of N sub-networks $\{\boldsymbol{o}_i\}_{i=1,2,...,N}$, $\boldsymbol{o}_i = [x_{i1}, x_{i2}, ..., x_{ic}] \in \mathbb{R}^C$.

Voting Voting uses mode value as the final output. If more than one mode exist, randomly choose one of them.

$$\boldsymbol{o}_{mode} = Mode(\boldsymbol{o}_1, \boldsymbol{o}_2, ..., \boldsymbol{o}_N)$$

Averaging The final output is the mean value over softmax outputs.

$$\boldsymbol{o}_{avg} = \frac{1}{N}\sum_{i=0}^{N} \boldsymbol{o}_i$$

Stacking A fully connected layer is used to gather output information from all N outputs together to generate final output. That means information from different classes will have slight effects on each other. In this kind, additional parameter size is NC^2.

$$\boldsymbol{o}_{stacking} = \boldsymbol{O} \cdot \boldsymbol{W}$$

$$\boldsymbol{O} = \begin{bmatrix} o_1 & o_2 & ... & o_N \end{bmatrix} \in \mathbb{R}^{1\times NC}, \boldsymbol{W} \in \mathbb{R}^{NC\times C}$$

Averaging and voting are parameter-free methods and do not need extra training. While benefit by supervised information, stacking can achieve slightly better accuracy in most cases, with marginal parameters added. So we mainly report the performance of intra-ensemble with random cut and stacking.

5. Experiments and Results

Preliminary In the experiment part, we carry out intra-ensemble on three series of networks, ResNet [11], ShuffleNet [37] and MobileNet [39]. We mainly focus on the experiment with a modified MobileNet V2 1.0x because of its high performance. Moreover, extensive experiments are implemented on various kinds of classification tasks to show the applicability and adaptability of intra-ensemble. The number of sub-networks is critical and should be set prudently. As sub-network number increases, conflict problem gradually emerges. Obviously, large amounts of sub-networks will result in a serious accuracy drop. According to our practice, 4 or 5 sub-networks is suitable for intra-ensemble on classification tasks. Under this fixed setting, sub-networks maintain high accuracy and diversity while intra-ensemble outperforms the sub-networks by a significant margin.

5.1. Results on CIFAR-10, CIFAR-100 [40]

Training setup We use PyTorch [41] for all our experiments. Our training setup follows the CIFAR-10 implementation of DARTS [28]. With smaller parameter size, a batch size of 128 can be used to train all our networks. Data augmentation is exactly the same as DARTS with cutout [42]. Moreover, we do not add any auxiliary head to assist in training.

Method	Parameters	Error(%)
ResNet [11]	1.7M	6.61
DenseNet(k = 24) [12]	27.2M	3.74
DenseNet-BC(k = 40) [12]	25.6M	3.46
Shake-Shake 26 2x96d [43]	26.2M	2.56
NASNet-A [44]	3.3M	2.65
NASNet-B [44]	2.6M	3.73
NASNet-C [44]	3.1M	3.59
PNASNet-5 [45]	3.2M	3.41
AmoebaNet-A [46]	3.2M	3.34
AmoebaNet-B [46]	2.8M	2.55
DARTS [28]	3.4M	2.83
ILRAS [47]	3.91M	2.60
IENet-A(4 sub-nets, RC)	1.63M	2.91
IENet-B(4 sub-nets, RC)	**2.66M**	2.61
IENet-C(4 sub-nets, RC)	4.22M	**2.47**

Table 1: Comparison of test error on CIFAR-10.

CIFAR-10 The configuration for CIFAR-10 experiments is: 4 sub-nets with random cut(RC) and a width ratio list $[0.9, 0.9, 0.9, 1.0]$. It reaches 2.61% test error, using only 2.66M parameters. With less parameters, it outperforms most neural architecture searched and manually designed models. Moreover, we have a wider version with 4.22M reaching 2.47% test error and a narrower version

Dataset	Operation	Sub-networks Acc.				Similarity $\mathcal{S}$	Intra-Ensemble Acc.	Δ
CIFAR-10	slimmable [15]	96.34	96.49	96.54	96.55	0.873	96.57	+0.02
	random cut & depth	96.35	96.41	96.76	96.57	0.863	97.09	+0.52
	random cut	96.70	96.79	96.71	96.68	0.857	97.39	+0.60
	random offset	96.49	96.31	96.40	96.42	0.845	97.23	+0.74
	shuffle channel	95.71	96.00	95.87	95.99	0.838	97.14	+1.14
CIFAR-100	slimmable [15]	79.72	80.21	80.77	81.42	0.807	81.70	+0.28
	random cut	80.70	80.79	80.74	80.87	0.759	82.98	+2.11
	random offset	78.45	78.65	78.95	78.89	0.685	82.90	+3.95
	shuffle channel	77.16	77.62	77.24	76.89	0.653	81.96	+4.34

Table 2: Intra-Ensemble details on CIFAR-10, CIFAR-100. All experiments are implemented using 4 sub-networks IENet-B with stacking. Δ means the accuracy difference between best sub-network and intra-ensemble.

with 1.63M reaching 2.91%, which proves the high scalability of intra-ensemble. The comparison with our models and others can be found in table 1. All the models with intra-ensemble have great improvement in accuracy while introducing little extra parameters. Notably, our 2.57M stand-alone network attains 3.10% test error, which is already comparable with NAS results. The baseline network architecture is a simply modified MobileNet V2 1.0x with slightly wider channels in each layer.

Method	Parameters	Error(%)
Wide ResNet [48]	11M	22.07
DenseNet-BC(k=40) [12]	25.6M	17.18
MetaQNN [49]	11.2M	27.14
SMASHv2 [27]	16M	20.6
Block-QNN-S [50]	6.1M	20.65
ENAS [31]	4.6M	17.27
PNAS [45]	3.2M	17.63
AmoebaNet-B [46]	34.9M	15.80
IENet(stand-alone)	2.71M	18.66
IENet(4 sub-nets, RC)	**2.78M**	**17.02**
IENet(5 sub-nets, RC)	**2.82M**	**16.35**

Table 3: Comparison of test error on CIFAR-100.

CIFAR-100 We directly apply CIFAR-10 configuration to CIFAR-100, except the output classes number. Though class number increases from 10 to 100 and bad cases become more complex, our intra-ensemble still have significant effects on improving performance. The 4 sub-nets IENet with 2.78M parameters have a 1.23% marginal gain compared to single model with similar parameter size, as in table 3. Surprisingly, the 5 sub-nets IENet with 2.82M parameters have a 2.31% marginal gain. We conjecture it is due to the relatively low top-1 accuracy compared with CIFAR-10. So intra-ensemble has more room for improvement.

Combination strategy As shown in table 4, averaging and stacking have higher accuracy than voting, which also happens in all other experiments. We conjecture that voting loses exact numerical information when converting output float values to class indexes. Because stacking usually has a slightly better result, we mainly report stacking performance in other experiments by default.

Strategy	CIFAR-10 Err.(%)	CIFAR-100 Err.(%)
Stand-alone	3.1	18.66
Voting	2.9	16.98
Averaging	2.61	16.37
Stacking	**2.61**	**16.35**

Table 4: Results of different combination strategies on CIFAR-10 and CIFAR-100.

Similarity analysis A good trade-off between accuracy and similarity should be carefully designed. As shown in table 2, these stochastic training operations significantly decrease $\mathcal{S}$. We can see different accuracy gaps when using different operations. When $\mathcal{S}$ is high, such as using slimmable or depth operation, the improvement by intra-ensemble is marginal. However, when $\mathcal{S}$ is low, such as using shuffle channel, the performance of sub-networks is affected by the disorder of channels. Although having small similarity, the intra-ensemble result using SC can not surpass results using RC or RO. Though the accuracy of sub-networks have a slight drop, intra-ensemble still outperforms stand-alone model easily with almost same parameter size. Moreover, the similarity $\mathcal{S}$ on CIFAR-100 is much lower than that in CIFAR-10. Correspondingly, intra-ensemble result has more improvement on CIFAR-100. We deduce that the complexity and variety of CIFAR-100 lead to more difference among sub-networks, thus ensemble performance is enhanced. Besides, in table 5, the datasets with more classes also have larger improvement with intra-ensemble. Presumably, intra-ensemble works better when dealing with more complex tasks.

Datasets	Classes	Input size	Dataset size	Baseline acc.	IENet acc.	Acc. Δ	Param Δ
SVHN [51]	10	32×32	99089	97.59	**97.93**	+0.34	+0.09M
Fashion-MNIST [52]	10	32×32	70000	96.05	**96.43**	+0.38	+0.09M
Mini-ImageNet [53]	100	84×84	60000	72.43	**74.71**	+2.28	+0.11M
Oxford-IIIT Pets [54]	37	224×224	7349	92.86	**94.91**	+2.05	+0.11M
FGVC Aircraft(Fam.) [55]	70	224×224	10200	82.60	**87.22**	+4.62	+0.11M
FGVC Aircraft(Var.) [55]	102	224×224	10200	75.34	**80.91**	+5.57	+0.11M
Caltech-101 [56]	101	224×224	9146	84.50	**87.65**	+3.15	+0.11M
Food-101 [57]	101	224×224	101000	82.27	**85.00**	+2.73	+0.11M

Table 5: Results on other datasets. All experiments are implemented using 4 sub-networks IENet-B with random cut and stacking. **Acc.** Δ means the accuracy difference between IENet and the stand-alone baseline network. **Param** Δ means the parameters added with intra-ensemble.

5.2. Additional Experiments

Other datasets We additionally carry out sufficient experiments on different kinds of datasets to prove the solidity of our method as shown in table 5. For the datasets without original division, training set and test set are randomly divided by a proportion of 9:1. When training models on these datasets, we do not spend much time on adjusting augmentations and hyper-parameters to pursue the best results. Simple training skills and fewer training epochs are applied in order to quickly verify the effectiveness of intra-ensemble on these datasets. All the extra experiments are carried out on 4 sub-networks IENet-B using random cut and stacking. While different datasets have various image sizes and class numbers, intra-ensemble always gains performance improvement on both simple and complex tasks. The results in the table demonstrate the general applicability of intra-ensemble to various types of data.

Other network architectures We implement intra-ensemble method on ShufflenetV2 1.0x [37] and ResNet34 in CIFAR-10 as well. These two and MobileNetV2 are excellent human-designed architectures with popular manually designed basic units, like residual block [11], depthwise-separable convolution [58], group convolution [59] and feature shuffling [60]. The size of ShuffleNetV2 1.0x is about 1.73M while for ResNet34 it is about 21.3M. Results can be found in table 6. Performance gap is not so large because of the accuracy saturation on CIFAR-10. Even so, results of intra-ensemble surpass their stand-alone trained ones with comparable parameter size. Experiments on these representative models prove that our method can be well applied to different types of networks.

Model	Original Err.	IENet Err.
ResNet34 [11]	3.84	**3.65**
ShuffleNetV2 1.0x [37]	5.47	**5.29**

Table 6: Results of other architectures on CIFAR-10.

5.3. Ablation Studies

Accuracy limitation of single network Commonly, deeper and wider network with more blocks and channels leads to better performance in accuracy. However, there often exists various kinds of hard examples in a certain dataset. One single neural network trained on training dataset usually ends at a local convergence state, which can hardly fit the whole test dataset. On this occasion, one neural network with limited parameters fail to solve all the hard cases. At the same time, increasing network size has very little marginal gain when parameters are already saturate on certain dataset.

Model	Params	Test Error (%)
stand-alone	0.48M	4.16
	1.56M	3.25
	2.57M	3.10
	3.41M	2.99
	4.11M	2.94
	4.73M	2.93
	6.01M	2.94
	8.38M	2.98
ensemble	2.57M×4	2.59
intra-ensemble	**1.63M**	**2.91**
	2.66M	**2.61**
	4.22M	**2.47**

Table 7: Stand-alone network performance and intra-ensemble of 4 sub-networks on CIFAR-10. In addition, ensemble of 4 indenpendent networks has similar result as intra-ensemble while using multiple parameters.

In table 7, as parameter size gradually increases, test accuracy saturates at a relative high level and it is hard to make a further improvement. However, with several independently trained small networks, ensemble easily achieves a better result which is almost impossible for stacking more layers or widening the channels. Besides, we run an ex-

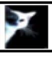
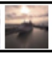
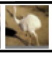
	err.																				
sub-network 1	3.30	✓	✗	✓	✗	✓	✗	✓	✓	✗	✓	✓	✗	✗	✗	✓	✓	✗	✓	✗	✓
sub-network 2	3.21	✗	✓	✓	✓	✓	✓	✗	✗	✓	✗	✗	✗	✓	✓	✓	✓	✗	✓	✗	✗
sub-network 3	3.29	✓	✓	✓	✗	✗	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓	✗	✓	✓	✓	✗
sub-network 4	3.32	✗	✓	✗	✓	✓	✗	✓	✓	✗	✗	✗	✓	✓	✓	✗	✓	✓	✗	✓	✓
stand-alone	3.10	✗	✓	✗	✓	✗	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	✗	✓	✓	✗
intra-ensemble	2.61	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

Figure 5: Visualization on CIFAR-10 dataset.

periment which simply applies ensemble to four individually trained networks with same architecture. Such experiment achieves similar result as intra-ensemble but using four times parameters. Our experiment results show that intra-ensemble has similar performance while using far less parameters compared with the simple one.

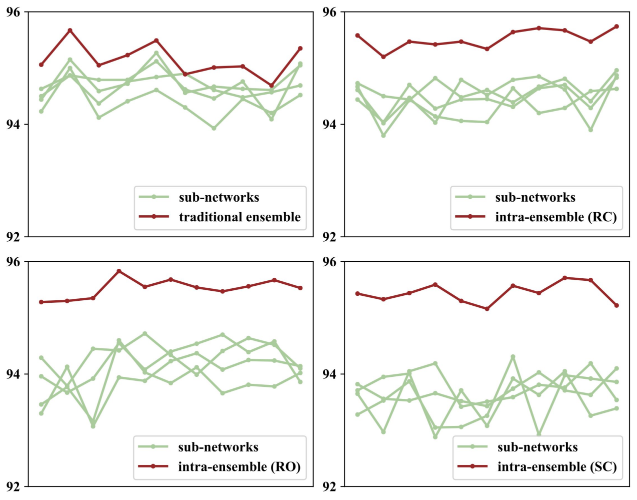


Figure 6: Accuracy gap between sub-networks and ensemble on CIFAR-10. (a) normal ensemble with slimmable neural network. (b) intra-ensemble with random cut. (c) intra-ensemble with random offset. (d) intra-ensemble with shuffle channel.

Naive training leads to homogenization One reason for the high performance of ensemble is that, different models have various preference and co-decision is statistically better. For a certain hard example, one model may fail. But when combining several independent models, if most outputs are correct, the final decision is correct as well. More intuitively, We draw the test accuracy curve of 10 continuous epochs from halfway through training. At first, we try to train original slimmable neural networks [15] or skip block on depth with normal ensemble techniques. But results in table 2 and figure 6 show that normal ensemble with slimmable operations has similar accuracy compared to sub-networks. Worse still, if one sub-network has a relatively low accuracy, it will markedly harm ensemble performance and even result in lower accuracy than the best sub-network. While in our case, greater diversity among sub-networks can be created by stochastic training methods. Therefore, intra-ensemble produces diversity among sub-networks, which ensures considerable improvement of model accuracy by ensemble learning.

5.4. Visualization

To show the improvement by intra-ensemble intuitively, we compare the test result of sub-networks, stand-alone network and intra-ensemble result on CIFAR-10. The pictures are randomly chosen from the test dataset if any sub-network has an incorrect output. As shown in figure 5, each sub-network respectively makes mistakes in several different images. Even stand-alone network cannot achieve perfect result because it presumably overfits part of the dataset. While with intra-ensemble, co-decision has the ability to eliminate more bad cases.

6. Conclusion

We have introduced Intra-Ensemble, a novel strategy which combines several diversified sub-networks within one neural network to enhance final performance. The stochastic training operations ensure high-accuracy and diversity of them. With marginal parameters added, intra-ensemble achieves competitive results compared with neural architecture search methods on classification tasks. Extensive experiments also show that our method is effective on various kinds of architectures and datasets. Besides, as multi-core computing power are more and more widespread, model parallelism will become more easily. Also, more techniques to improve training efficiency like shuffling data could be explored. On this occasion, our IENets can produce outstanding result with very limited occupation of storage. In this work, we only carry out experiments on classification tasks. It is obvious that future work on other computer vision tasks (e.g. segmentation, object detection) can be explored to verify the transferability of intra-ensemble. We will work on it to maximize the utilization of intra-ensemble method.

References

[1] Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[2] Yoav Freund, Robert Schapire, and Naoki Abe. A short introduction to boosting. *Journal-Japanese Society For Artificial Intelligence*, 14(771-780):1612, 1999.

[3] Zenglin Shi, Le Zhang, Yun Liu, Xiaofeng Cao, Yangdong Ye, Ming-Ming Cheng, and Guoyan Zheng. Crowd counting with deep negative correlation learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5382–5390, 2018.

[4] Thomas Elsken, Jan-Hendrik Metzen, and Frank Hutter. Simple and efficient architecture search for convolutional neural networks. *arXiv preprint arXiv:1711.04528*, 2017.

[5] Cheng Ju, Aurélien Bibaut, and Mark van der Laan. The relative performance of ensemble methods with deep convolutional neural networks for image classification. *Journal of Applied Statistics*, 45(15):2800–2818, 2018.

[6] Shu Liu, Lu Qi, Haifang Qin, Jianping Shi, and Jiaya Jia. Path aggregation network for instance segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8759–8768, 2018.

[7] Li Wan, Matthew Zeiler, Sixin Zhang, Yann Le Cun, and Rob Fergus. Regularization of neural networks using dropconnect. In *International conference on machine learning*, pages 1058–1066, 2013.

[8] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alexander A Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[9] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.

[10] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826, 2016.

[11] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[12] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708, 2017.

[13] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7132–7141, 2018.

[14] Gabriel Bender, Pieter-Jan Kindermans, Barret Zoph, Vijay Vasudevan, and Quoc Le. Understanding and simplifying one-shot architecture search. In *International Conference on Machine Learning*, pages 549–558, 2018.

[15] Jiahui Yu, Linjie Yang, Ning Xu, Jianchao Yang, and Thomas Huang. Slimmable neural networks. *arXiv preprint arXiv:1812.08928*, 2018.

[16] Yong Liu and Xin Yao. Ensemble learning via negative correlation. *Neural networks*, 12(10):1399–1404, 1999.

[17] Rich Caruana, Alexandru Niculescu-Mizil, Geoff Crew, and Alex Ksikes. Ensemble selection from libraries of models. In *Proceedings of the twenty-first international conference on Machine learning*, page 18. ACM, 2004.

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

[19] Thomas G Dietterich et al. Ensemble learning. *The handbook of brain theory and neural networks*, 2:110–125, 2002.

[20] Zhi-Hua Zhou. *Ensemble methods: foundations and algorithms*. Chapman and Hall/CRC, 2012.

[21] Leo Breiman. Bagging predictors. *Machine learning*, 24(2):123–140, 1996.

[22] Yoav Freund, Robert E Schapire, et al. Experiments with a new boosting algorithm. In *icml*, volume 96, pages 148–156. Citeseer, 1996.

[23] David Wolpert and William G Macready. Combining stacking with bagging to improve a learning algorithm. *Santa Fe Institute, Technical Report*, 1996.

[24] Thomas Elsken, Jan Hendrik Metzen, and Frank Hutter. Neural architecture search: A survey. *arXiv preprint arXiv:1808.05377*, 2018.

[25] Esteban Real, Sherry Moore, Andrew Selle, Saurabh Saxena, Yutaka Leon Suematsu, Jie Tan, Quoc Le, and Alex Kurakin. Large-scale evolution of image classifiers. *arXiv preprint arXiv:1703.01041*, 2017.

[26] Barret Zoph and Quoc V Le. Neural architecture search with reinforcement learning. *arXiv preprint arXiv:1611.01578*, 2016.

[27] Andrew Brock, Theodore Lim, James M Ritchie, and Nick Weston. Smash: one-shot model architecture search through hypernetworks. *arXiv preprint arXiv:1708.05344*, 2017.

[28] Hanxiao Liu, Karen Simonyan, and Yiming Yang. Darts: Differentiable architecture search. *arXiv preprint arXiv:1806.09055*, 2018.

[29] Han Cai, Ligeng Zhu, and Song Han. Proxylessnas: Direct neural architecture search on target task and hardware. *arXiv preprint arXiv:1812.00332*, 2018.

[30] Bichen Wu, Xiaoliang Dai, Peizhao Zhang, Yanghan Wang, Fei Sun, Yiming Wu, Yuandong Tian, Peter Vajda, Yangqing

Jia, and Kurt Keutzer. Fbnet: Hardware-aware efficient convnet design via differentiable neural architecture search. *arXiv preprint arXiv:1812.03443*, 2018.

[31] Hieu Pham, Melody Y Guan, Barret Zoph, Quoc V Le, and Jeff Dean. Efficient neural architecture search via parameter sharing. *arXiv preprint arXiv:1802.03268*, 2018.

[32] Juan-Manuel Pérez-Rúa, Moez Baccouche, and Stephane Pateux. Efficient progressive neural architecture search. *arXiv preprint arXiv:1808.00391*, 2018.

[33] Tianqi Chen, Ian Goodfellow, and Jonathon Shlens. Net2net: Accelerating learning via knowledge transfer. *arXiv preprint arXiv:1511.05641*, 2015.

[34] Tao Wei, Changhu Wang, Yong Rui, and Chang Wen Chen. Network morphism. In *International Conference on Machine Learning*, pages 564–572, 2016.

[35] Hokchhay Tann, Soheil Hashemi, R Bahar, and Sherief Reda. Runtime configurable deep neural networks for energy-accuracy trade-off. In *Proceedings of the Eleventh IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis*, page 34. ACM, 2016.

[36] Roxana Istrate, Adelmo Cristiano Innocenza Malossi, Costas Bekas, and Dimitrios Nikolopoulos. Incremental training of deep convolutional neural networks. *arXiv preprint arXiv:1803.10232*, 2018.

[37] Ningning Ma, Xiangyu Zhang, Hai-Tao Zheng, and Jian Sun. Shufflenet v2: Practical guidelines for efficient cnn architecture design. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 116–131, 2018.

[38] John S Bridle. Training stochastic model recognition algorithms as networks can lead to maximum mutual information estimation of parameters. In *Advances in neural information processing systems*, pages 211–217, 1990.

[39] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4510–4520, 2018.

[40] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009.

[41] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017.

[42] Terrance DeVries and Graham W Taylor. Improved regularization of convolutional neural networks with cutout. *arXiv preprint arXiv:1708.04552*, 2017.

[43] Xavier Gastaldi. Shake-shake regularization. *arXiv preprint arXiv:1705.07485*, 2017.

[44] Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V Le. Learning transferable architectures for scalable image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8697–8710, 2018.

[45] Chenxi Liu, Barret Zoph, Maxim Neumann, Jonathon Shlens, Wei Hua, Li-Jia Li, Li Fei-Fei, Alan Yuille, Jonathan Huang, and Kevin Murphy. Progressive neural architecture search. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 19–34, 2018.

[46] Esteban Real, Alok Aggarwal, Yanping Huang, and Quoc V Le. Regularized evolution for image classifier architecture search. *arXiv preprint arXiv:1802.01548*, 2018.

[47] Minghao Guo, Zhao Zhong, Wei Wu, Dahua Lin, and Junjie Yan. Irlas: Inverse reinforcement learning for architecture search. *arXiv preprint arXiv:1812.05285*, 2018.

[48] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. *arXiv preprint arXiv:1605.07146*, 2016.

[49] Bowen Baker, Otkrist Gupta, Nikhil Naik, and Ramesh Raskar. Designing neural network architectures using reinforcement learning. *arXiv preprint arXiv:1611.02167*, 2016.

[50] Zhao Zhong, Junjie Yan, Wei Wu, Jing Shao, and Cheng-Lin Liu. Practical block-wise neural network architecture generation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2423–2432, 2018.

[51] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng. Reading digits in natural images with unsupervised feature learning. 2011.

[52] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

[53] Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Daan Wierstra, et al. Matching networks for one shot learning. In *Advances in neural information processing systems*, pages 3630–3638, 2016.

[54] Yan Em, Feng Gag, Yihang Lou, Shiqi Wang, Tiejun Huang, and Ling-Yu Duan. Incorporating intra-class variance to fine-grained visual recognition. In *2017 IEEE International Conference on Multimedia and Expo (ICME)*, pages 1452–1457. IEEE, 2017.

[55] Subhransu Maji, Esa Rahtu, Juho Kannala, Matthew Blaschko, and Andrea Vedaldi. Fine-grained visual classification of aircraft. *arXiv preprint arXiv:1306.5151*, 2013.

[56] Li Fei-Fei, Rob Fergus, and Pietro Perona. Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories. *Computer vision and Image understanding*, 106(1):59–70, 2007.

[57] Lukas Bossard, Matthieu Guillaumin, and Luc Van Gool. Food-101 – mining discriminative components with random forests. In *European Conference on Computer Vision*, 2014.

[58] François Chollet. Xception: Deep learning with depthwise separable convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1251–1258, 2017.

[59] Saining Xie, Ross Girshick, Piotr Dollar, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[60] Xiangyu Zhang, Xinyu Zhou, Mengxiao Lin, and Jian Sun. Shufflenet: An extremely efficient convolutional neural network for mobile devices. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6848–6856, 2018.