"In-Network Ensemble": Deep Ensemble Learning with Diversified Knowledge Distillation

XINGJIAN LI, Baidu Inc., China and University of Macau, China
HAOYI XIONG, ZEYU CHEN, and JUN HUAN, Baidu Inc., China
CHENG-ZHONG XU, University of Macau, China
DEJING DOU, Baidu Inc., China

Ensemble learning is a widely used technique to train deep convolutional neural networks (CNNs) for improved robustness and accuracy. While existing algorithms usually first train multiple diversified networks and then assemble these networks as an aggregated classifier, we propose a novel learning paradigm, namely, *"In-Network Ensemble"* (**INE**) that incorporates the diversity of multiple models through training a SINGLE deep neural network. Specifically, **INE** segments the outputs of the CNN into multiple independent classifiers, where each classifier is further fine-tuned with better accuracy through a so-called *diversified knowledge distillation process*. We then aggregate the fine-tuned independent classifiers using an Averaging-and-Softmax operator to obtain the final ensemble classifier. Note that, in the supervised learning settings, **INE** starts the CNN training from random, while, under the transfer learning settings, it also could start with a pre-trained model to incorporate the knowledge learned from additional datasets. Extensive experiments have been done using eight large-scale real-world datasets, including CIFAR, ImageNet, and Stanford Cars, among others, as well as common deep network architectures such as VGG, ResNet, and Wide ResNet. We have evaluated the method under two tasks: supervised learning and transfer learning. The results show that **INE** outperforms the state-of-the-art algorithms for deep ensemble learning with improved accuracy.

CCS Concepts: • **Computing methodologies** → **Ensemble methods**; **Transfer learning**; *Neural networks;*

Additional Key Words and Phrases: Transfer learning, knowledge distillation, ensemble learning

ACM Reference format:
Xingjian Li, Haoyi Xiong, Zeyu Chen, Jun Huan, Cheng-Zhong Xu, and Dejing Dou. 2021. "In-Network Ensemble": Deep Ensemble Learning with Diversified Knowledge Distillation. *ACM Trans. Intell. Syst. Technol.* 12, 5, Article 63 (December 2021), 19 pages.
https://doi.org/10.1145/3473464

Xingjian Li and Haoyi Xiong contributed equally to the article.

This article is supported by National Key Research and Development Program of China (No. 2019YFB2102100), the Science and Technology Development Fund of Macau SAR (File no. 0015/201-9/AKP), GuangDong Basic and Applied Basic Research Foundation No. 2020B515130004, and Key-area Research and Development Program of Guangdong Province (No. 2020B010164003).

Authors' addresses: X. Li, Big Data Lab, Baidu Inc. Haidian, Beijing, 100193, China, Department of Computer and Information Sciences, University of Macau, Taipa, Macau, China; email: lixingjian@baidu.com; H. Xiong (corresponding author), J. Huan, and D. Dou, Big Data Lab, Baidu Inc. Haidian, Beijing, 100193, China; emails: xionghaoyi@baidu.com, huanjun@baidu.com, doudejing@baidu.com; Z. Chen, Deep Learning Technology Platform, Baidu Inc. Shenzhen, Guandong, 518000, China; C.-Z. Xu, Department of Computer and Information Sciences, University of Macau, Taipa, Macau, China; email: czxu@um.edu.mo.

2157-6904/2021/12-ART63 $15.00
https://doi.org/10.1145/3473464

1 INTRODUCTION

Deep neural networks are frequently trained with large variance [11], where such networks overfit to the noise in the training dataset and produce low generalizability in reality. To reduce the variance, ensemble learning that aggregates the power of multiple "base" models has been widely used to improve the performance of deep learning with enhanced generalizability [12, 26]. Due to the time-consuming training of deep neural networks, training multiple deep models independently may take a prohibitively large amount of time and/or computational resources and hence hinders the wide application of deep ensemble learning. There is a pressing need to produce the ensemble of multiple deep models efficiently (e.g., there is an upper-limit of computational time or resource of training) as argued in a few very recent studies [8, 10, 19, 21, 43].

Recent studies, including Snapshot Ensemble [19], Temporal Ensemble [27], and Mean Teacher [45], also propose to ensemble multiple models sampled from the training trajectory of a single deep neural network. Similarly, researchers in Reference [10] propose **Fast Geometric Ensemble (FGE)**, which utilizes the geometry of landscape near the local minima and improves the model sampling over the training trajectories. The authors in Reference [21] further improve FGE using the **Stochastic Weight Averaging algorithm (SWA)**, which aggregates the weights of multiple snapshots through a **Bayesian parameter averaging (BPA)** scheme. Although these methods are capable of obtaining diverse models efficiently, most of them (except for SWA) require much more storage and calculation when performing prediction, as the multiple models are used for the inference.

Different from existing methods rather than considering snapshots from the training trajectory, our work is motivated by the fact that deep neural networks are highly over-parameterized while parameters in many neurons are redundant [1, 34]. Many studies have shown that a large proportion of the DNN's parameters can be pruned without significant performance reduction, such as References [7, 29]. In this way, we tend to improve the performance of a DNN model by further developing the redundant parts of DNN beyond the convergence of **empirical risk minimization (ERM)**. Though fewer parameters are incorporated, the student learners are still capable of achieving comparable accuracy, due to the excellent representations learned through knowledge distillation from the teacher model. Specifically, we propose a spatial-temporal ensemble paradigm, ***"In-Network Ensemble"*** **(INE)**, that spatially subdivides a trained classifier into multiple parts, trains each part with a knowledge distillation method [17], and aggregates the base classifiers from the spatial domain. The paradigm is uniquely designed to encourage diversity among temporal classifiers and to facilitate the fine-tuning of each individual classifier for the better classification accuracy.

Specifically, given a deep network architecture (e.g., VGGs or ResNets) and a training dataset, **INE** first trains a network using the dataset based on the architecture. Note that, in this period, the network can be trained using either common optimizers such as **Stochastic Gradient Descent (SGD)** [28], or other ensemble trainers [10, 21] or even transfer learning trainer [31, 49]. With the pre-trained model as the teacher network, the algorithm duplicates the weights of teacher network as the initialization of a student network. Then, **INE** segments the fully connected layers of the student network into multiple individual classifiers, where each individual classifier outputs the classification results using the shared convolutional layers and its own part of fully connected layers. With **INE**, every individual classifier in the student network is further fine-tuned for better classification accuracy while preserving the diversity. To accelerate the fine-tuning procedure with better accuracy, knowledge distillation via Teacher-Student training [17] has been used to align the output vector between the teacher network (i.e., pre-trained model) and the fined-tuning one.

To the end, these fine-tuned classifiers in the same network are aggregated using an Averaging-and-Softmax operator (such as Reference [19]) as an ensemble classifier.

Our Contributions. To enable computation-efficient deep ensemble learning, the proposed **INE** made contributions in both neural architectures and training algorithms as detailed below. First, in terms of architecture, compared to traditional approaches [8, 10, 19, 43], which either aggregate multiple models of different architectures or average weights/outputs of a specific model at different training snapshots, **INE** *"physically"* subdivides a pre-trained neural network into multiple learners for the same task, then averages the prediction these learners as the overall classifier. Second, in terms of training algorithms, **INE** is the first to leverage knowledge distillation [17] to improve the training of individual learners to boost the overall ensemble performance. To increase the diversity among base learners, we have designed a novel diversified Teacher-Student training algorithm, which significantly improves the performance of ensemble learning. Third, we find that **INE** has a natural application in deep transfer learning with additional performance boost caused by ensemble learning. To the best of our knowledge, this work is the first study to address the neural architectures, knowledge distillation via teacher-student training, ensemble of multiple individual classifiers with shared convolutional layers, the number of parameters, and computational time/resource budget issues in deep ensemble learning research.

We have performed extensive experiments using eight large-scale real-world datasets, including ImageNet and Stanford Cars, among others, as well as the common deep network architectures such as VGG, ResNet, and Wide ResNet, in both supervised learning and transfer learning settings. The results show **INE** outperforms the state-of-the-art algorithms for deep ensemble learning with higher accuracy. Furthermore, under transfer learning settings, we find that **INE** leads to additional performance boosts on top of the state-of-the-art transfer learning algorithms [31].

Organization of the Article. The rest of this article is organized as follows: Section 2 briefs the preliminary work and the technical backgrounds of our work, where the state-of-the-art algorithms for the deep ensemble learning are introduced with details. Section 3 presents the design of the proposed algorithm **INE**, where we first propose overall framework, then present the detailed algorithm design. In Section 4, we report the experiments that we conducted to evaluate **INE**, where we first present the experiment setups and datasets used, then introduce the main results with the accuracy comparisons between **INE** and the existing deep ensemble learning algorithms; further, we provide a case study where we evaluate the performance of **INE** under the transfer learning settings. Finally, we review the related work, discuss open issues, and conclude this work in Sections 5 and 6.

2 NOTATIONS AND PRELIMINARIES

In this section, we review the existing algorithms for ensemble learning and knowledge distillation, where we specify the notations and the design of algorithms and their connections to **INE**. The notations used in the manuscript have been listed in Table 1 for clear demonstration.

2.1 Ensemble Learning via Model Averaging

Averaging multiple models as an ensemble classifier was originally used in statistical learning paradigms [9]. Given a dataset $\mathcal{D} = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \ldots, (\boldsymbol{x}_n, y_n)\}$ for supervised learning and a classification model $f(\boldsymbol{x}|\theta) \rightarrow y$ that maps input $\boldsymbol{x}$ to label y based on the parameter $\theta \in \Theta \subseteq \mathbb{R}^k$ (the integer k has been already defined in Section 1 as the number of parameters in a neural network), ensemble learning algorithms usually first learn a set of M parameters $\{\theta_1, \theta_2, \ldots, \theta_M\} \subseteq \Theta$, then aggregate these parameters as an overall model. Basically, we summarize model aggregation mechanisms as follows:

Table 1. List of Symbols

Symbol	Description
$\mathcal{D}$	The training dataset for supervised learning. For transfer learning it refers in particular to the target dataset.
$\boldsymbol{x}$	The input of an example.
y	The label of an example.
f	The function of the deep neural network.
θ	The parameter of the deep neural network.
Θ	Domain of the parameter θ.
k	The dimension of the parameter θ.
M	The number of ensembled parameters or models.
L	The loss function corresponding to Empirical Risk Minimization.
Ω	The regularization function.
λ	The coefficient of the regularization function Ω.

- *Parameter Averaging.* The most recent work *SWA* [21] proposed to use the classifier $f(\boldsymbol{x}|\widehat{\theta})$ with an averaged parameter $\widehat{\theta}$, where

$$\widehat{\theta} \leftarrow \frac{1}{M}\sum_{j=1}^{M}\omega_j \cdot \theta_j, \tag{1}$$

  and $\omega_1, \omega_2, \ldots, \omega_M$ refer to a set of weights (obtained through an adaptive scheme [21] during an SGD procedure) for parameter averaging. As was mentioned, above method provides a ensemble classifier with k-parameters based on M models (each with k-parameters).
- *Prediction Average.* The earlier work including snapshots ensemble [19], temporal ensembling [8, 27], and *FGE* [10] proposed to use an averaged classifier $\widehat{f}(\boldsymbol{x}|\theta_1, \theta_2, \ldots, \theta_M) \rightarrow y$, such that

$$\widehat{f}(\boldsymbol{x}|\theta_1, \theta_2, \ldots, \theta_M) \leftarrow \frac{1}{M}\sum_{j=1}^{M}\omega_j \cdot f(\boldsymbol{x}|\theta_j), \tag{2}$$

  and $\omega_1, \omega_2, \ldots, \omega_M$ refer to a set of weights for prediction results averaging. Compared to *parameter averaging*, above algorithms [8, 10, 19, 27] obtain an ensemble classifier with totally (kM)-parameters.

In our research, we adopt the *prediction averaging* strategy to aggregate classifiers. Compared to existing *prediction averaging* classifiers using kM parameters, we aim at providing a novel model with k-parameters but enjoying higher accuracy than parameter averaging model, i.e., SWA [21].

2.2 Knowledge Distillation

As was introduced in Section 1, knowledge distillation is a key technique used in our article. Given a pre-trained model $f^*(\boldsymbol{x})$ and the training dataset $\mathcal{D}$, knowledge distillation [17] technique aims at improving the training of a neural network with respect to the Logit output of $f^*(x)$ (denoted as $f^*_{\text{Logit}}(\boldsymbol{x})$) over the input space of $\mathcal{D}$. Specifically, with a empirical loss function $L(\theta)$, knowledge distillation employs a regularized loss function such that

$$\widehat{\theta}^* \leftarrow \operatorname*{argmin}_{\theta\in\Theta} L(\theta) + \lambda\Omega(\theta), \tag{3}$$

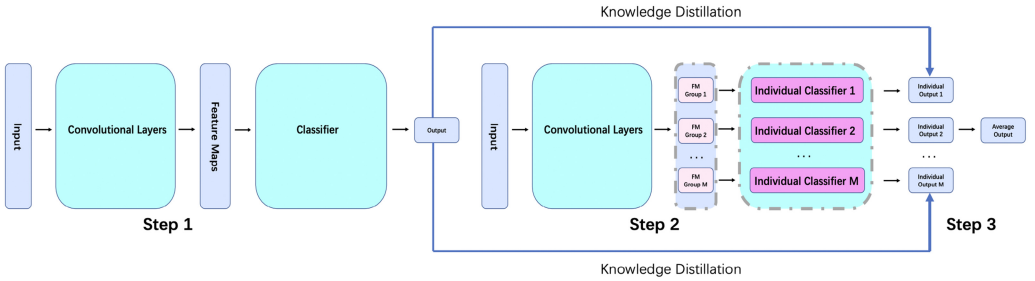

Fig. 1. The overall framework design of **INE**: **Step (1).** Pre-training a neural network (optionally using the dataset $\mathcal{D}$) as the teacher network and duplicating the pre-trained weights as the initialization of the student network; **Step (2).** Dividing the weights of fully connected layers of the student network into multiple individual classifiers and fine-tuning every individual classifier with the training dataset $\mathcal{D}$; and **Step (3.)** Averaging the logit outputs of all individual classifiers as the ensemble classifier through softmax. Note that, when using a much larger dataset than $\mathcal{D}$ to pre-train the teacher network, **INE** enables the transfer learning.

where λ refers to the coefficient of regularization and the regularization term $\Omega(\theta)$ is defined as

$$\Omega(\theta) = \frac{1}{n}\sum_{i=1}^{n} \|f^*_{\text{logit}}(\boldsymbol{x}_i) - f_{\text{logit}}(\boldsymbol{x}_i|\theta)\|_2^2. \tag{4}$$

Note that we define f_{logit} and f^*_{logit} as the logit outputs of the two neural networks, respectively, where Reference [17] assumed these outputs are with equal width. This mechanism was originally proposed to transfer knowledge learned in f^* to the parameter θ.

Specifically, f^* is not necessary to be trained using $\mathcal{D}$. In transfer learning set, the size of $\mathcal{D}$ was assumed to be small, Reference [49] proposed to use the pre-train mode f^* that has been pre-trained with a large dataset (such as ImageNet). This work uses knowledge distillation that transfers the knowledge learned by the pre-trained model to fine-tune every individual classifier and enable transfer learning (as a by-product) with **INE**.

3 INE: FRAMEWORK AND ALGORITHM DESIGN

We first present the overall framework design for **INE**and then discuss key algorithm components.

3.1 INE Overall Framework: Neural Architectures and Training Procedure

We illustrate the overall framework of "In-Network Ensemble" training with three steps in Figure 1 and provide the detailed description below.

- *Step (1).* Given the training dataset $\mathcal{D}$ and a specific neural network architecture, **INE** uses $\mathcal{D}$ to train a model based on the architecture.
- *Step (2).* With the pair of Teacher/Student networks, **INE** segments the fully connected layer of the Student network into M individual classifiers, where each classifier is based on the independent fully connected layer with shared convolutional layers (shown in Figure 1(b)). Supposing we have N channels in the last convolutional layer, corresponding to a N-dimensional feature vector F_v after the average pooling operator, we physically divide these N elements into M groups by their storage order. Specifically, the part $F_v[0 : N/M]$ is connected to the first base learner, $F_v[N/M : 2N/M]$is connected to the second base learner, and so on. Each individual classifier is randomly initialized using different seeds.
- *Step (3).* **INE** fine-tunes each of M classifiers using the training dataset $\mathcal{D}$ and the Teacher network. More specific, through knowledge distillation [17], **INE** performs a tight alignment

of the Logit outputs between the Teacher network and every individual classifier for performance boosts. Note that, to preserve the diversity among the classifiers, **INE** starts the training of every individual classifier from random scratch [10].

- *Step (4).* To aggregate the M individual classifiers as an ensemble classifier, **INE** adopts a simple "Average-and-Softmax" operator that averages the Logit outputs of the M individual classifiers and outputs the final classification result through softmax. In this way, **INE** obtains an ensemble classifier with k-parameters based on M models. Please refer to References [3, 9, 42] to understand the theoretical properties and generalization power of such averaging classifiers.

The overall computational time and resources used for **INE** can be broken down into two parts: the pre-training and fine-tuning procedures. To simplify the calculation, we estimate the overall complexity as the total number of epochs used by the two procedures, (the proportion of epochs spent for the first procedure is denoted as α, where the second procedure consumes $1-\alpha$ proportion of epochs). We would evaluate **INE** using various settings of α in Section 4.2.

In summary, we have demonstrated the overall framework design of **INE**, where the key algorithms specified for Teacher-Student training have not been well stated. We now proceed to present key algorithmic components in following section.

3.2 INE Training Algorithms

In this section, we introduce the Teacher-Student training algorithms design (Step (2)) for **INE**, where we first introduce the fine-tuning procedure given the pre-trained model, present the loss function designed for Teacher-Student training. We later show how **INE** works with the budgeted computational complexity and the fixed size of parameters. The key to the training dataset is a procedure for increasing the diversity of base learners, which we call the diversified Teacher-Student Training, detailed in this section as well.

ALGORITHM 1: **INE**: Training Algorithms

procedure INE($\mathcal{D}$)
  /*Obtain a Pre-trained Model*/
  $\theta^{pre} \leftarrow$ Obtain the weights through training with the dataset $\mathcal{D}$ (for supervised learning) or from other pre-trained models (for transfer learning)
  /*Initialize Fine-Tuning with Randomness*/
  $\theta_0 \leftarrow \theta^{pre}$ and Set the weights of fully connected layers with Gaussian random noise
  /*Fine-Tuning Procedure via (perturbed) Teacher-Student Training*/
  $\widehat{\theta}^M \leftarrow$ Run SGD to minimize the Loss function listed in Equation (5) from the starting point θ_0
  return $\widehat{\theta}^M$
end procedure

Fine-tuning Procedure via Diversified Teacher-Student Training. Given a pre-trained neural network $f(\boldsymbol{x}|\theta^{pre})$ with parameters θ^{pre}, **INE** assumes the Logit output of $f(\boldsymbol{x}|\theta^{pre})$ (denoted as $f_{\text{logit}}(\boldsymbol{x}_i|\theta^{pre})$) is with equal width to the Logit output of each individual classifier (denoted as $f^j_{\text{logit}}(\boldsymbol{x}_i|\theta)$ for the jth individual classifier based on parameters $\theta \in \Theta$). Then, **INE** fine-tunes the network with θ^{pre} using the algorithm listed in 1. Note that two mechanisms are employed to implement *diversified* Teacher-Student training. The first is to perform random initialization of different individual learners, which straightforwardly leads to diverse solutions. The second is

to incorporate random perturbations in Teacher network's Logit outputs to promote the divergence/diversity between classifiers.

Loss Function for Diversified Teacher-Student Training. Compared to knowledge distillation, **INE** employs a loss function with slightly different regularization. **INE** defines an operator $f^j_{\text{logit}}(x_i|\theta)$ refers to the Logit output of the jth individual classifier in the neural network based on the parameters θ and input x_i. On top of a common empirical loss function based on the training dataset $\mathcal{D}$ and a pre-trained model (obtained by Step (1)), **INE** employs a loss function such that

$$\widehat{\theta}^M \leftarrow \underset{\theta\in\Theta}{\text{argmin}}\, L(\theta) + \frac{\lambda}{M}\sum_{j=1}^{M}\widetilde{\Omega}^j(\theta), \tag{5}$$

where λ is the coefficient and the regularization term $\widetilde{\Omega}({}^j\theta)$ for $1 \le j \le M$ is defined as

$$\widetilde{\Omega}^j(\theta) = \frac{1}{n}\sum_{i=1}^{n}\|\widetilde{f}_{\text{logit}}(\boldsymbol{x}_i|\theta^{pre}) - f^j_{\text{logit}}(\boldsymbol{x}_i|\theta)\|_2^2. \tag{6}$$

Note that certain Squared-Euclidean distance regularizer is used in **INE** to constrain the search space of parameters from its starting point. Further, the parameters $\widehat{\theta}^M$ are used as the final training result. Specifically, $\widetilde{f}_{\text{logit}}(x_i|\theta^{pre})$ refers to the (noisy) Logit outputs of the Teacher network based on parameter θ^{pre} and input x_i. We define it as

$$\widetilde{f}_{\text{logit}}(\boldsymbol{x}_i|\theta^{pre}) = f_{\text{logit}}(\boldsymbol{x}_i|\theta^{pre}) + \mathcal{N}(0, \delta^2\mathbf{I}). \tag{7}$$

When $\delta = 0$, the term $\widetilde{f}_{\text{logit}}(\boldsymbol{x}_i|\theta^{pre})$ refers to the noiseless Logit outputs (i.e., the algorithm performs as the vanilla Teacher-Student networks training). When $\delta > 0$ the term provides Logit outputs with unbiased noise (i.e., the algorithm performs knowledge distillation from a perturbed teacher model).

We would like to point out that the perturbation added on the teacher model is optional, as the diversity among students learners naturally exists due to the random initialization. Our experiments found that the incorporation of unbiased noise in Logit outputs improves the performance for *"wider"* network training (such as WideResNet), where the total number of individual classifiers is relatively large and the diversity between classifiers should be concreted. To enable ensemble learning with a relatively *"narrow"* CNN, such as ResNet, the inclusion of noise may hurt the convergence of training procedure under the budgeted computational time without significant accuracy improvement.

4 EXPERIMENTS

In this section, we present the experimental results of **INE** for two groups of tasks: supervised learning tasks (pre-trained from the target training datasets) and transfer learning tasks (pre-trained from other datasets).

4.1 Datasets and Experiment Setups

We used a wide range of datasets summarized in Table 2 to evaluate the proposed algorithms. Specifically, we used CIFAR 10 and CIFAR 100 datasets for the experiments under ensemble learning settings. We had ImageNet as source dataset while using FGVC-Aircraft, Flowers 102, DTD and Stanford Cars datasets as targets for transfer learning settings. Following convention, we used fixed training and testing datasets, and the sizes of the datasets were provided in Table 2.

Table 2. Statistics on Datasets

Datasets	Domains	# Train/Test
FGVC_Aircraft	Aircraft	3.3K/3.3K
Stanford Cars	Cars	12K/8.5K
ImageNet	Ubiquitous Objects	1,419K+/100K
CIFAR 10	Ubiquitous Objects	50K/10K
CIFAR 100	Ubiquitous Objects	50K/10K
DTD	Describable Textures	1.8K/1.8K
Flowers 102	Flowers	1K+/6K+

4.1.1 Networks and Algorithms Settings. We evaluated our algorithms with three popular network architectures: VGG-16, ResNet-164, and WResNet-28-10 for ensemble learning tasks, where we compared **INE** with vanilla SGD, *SWA* [21], and FGE [10]. To perform a fair comparison, we focused on the classification accuracy improvement made by **INE** beyond each of these baselines with the same or even fewer number of parameters.

For transfer learning settings, we evaluated **INE** using ResNet-101 and Inception-V3, which are popular networks for transfer learning benchmarks. Furthermore, we compared **INE** with the state-of-the-art algorithms for deep transfer learning, such as L^2 [35] and L^2-SP [31], where we provided a fair evaluation of the performance of **INE** through comparison.

Hyperparameter choices. To perform fair comparison, we use the same setting of common hyperparameters such as weight decay and learning rate across all experiments according to baseline work [21]. For those hyperparameters specific to SWA [21] and FGE [10], we use exactly the same choice recommended in their papers. The only additional hyperparameter for **INE** is the coefficient λ, corresponding to the weight of regularization. We simply use the default value 1 for λ in all experiments involved **INE**.

SGD and SWA Settings. For vanilla SGD, we used the initial learning rate of 0.05 for VGG-16 and 0.1 for other architectures. The momentum of 0.9 is used for all architectures. We used the same learning rate schedule for all architectures described as the following: the learning rate is fixed to the initialization lr_0 for the first 50% of epochs, linearly decreased to $lr_0/10$ for the next 40% of epochs, and fixed to $lr_0/100$ for the remaining epochs. The Weight decay is set to $3e^{-4}$ for ResNet-164 and $5e^{-4}$ to other architectures. In SWA experiments, we used the constant learning rate 0.01 for VGG-16 and WResNet-28-10. For ResNet-164, we use the learning rate 0.01 on CIFAR-10 and 0.05 on CIFAR-100.

FGE Settings. In this work, we directly used the results of FGE method reported in Reference [10], where FGE is set to incorporate $M = 6 \sim 12$ snapshots (classifiers) as the ensemble classifier. As FGE leverages predictions averaging pattern, the parameter complexity is kM, i.e., with $6K \sim 12K$ parameters.

INE Settings. For ensemble learning tasks, we set $M = 2$—training the classifier that assembles two learners. Based on different pre-trained algorithms and budgets of parameter size, we validated three versions of **INE** comparing with the corresponding baseline algorithms.

- **INE:** Given certain computational budget for ensemble learning, this strategy first used SGD to pre-train the model with α (usually set to 0.8 to 0.9) budget, then spent the rest $(1 - \alpha)$ budget using **INE** to fine-tune the model.
- **INE+:** Given certain computational budget for ensemble learning, this strategy first used SWA to pre-train the model for α (usually set to 0.8 to 0.9) budget, then replaced the last $(1-\alpha)$ budget using **INE** to fine-tune the model. Compared to **INE**, **INE+** enjoyed additional performance boosts with ensemble learner based on pre-training. Note that for the SWA and

Table 3. Performance Comparison on Different Datasets

	CIFAR100			CIFAR10		
	VGG	ResNet	WResNet	VGG	ResNet	WResNet
SGD [21]	72.55 ± 0.10	78.49 ± 0.36	80.82 ± 0.23	93.25 ± 0.16	95.28 ± 0.10	96.18 ± 0.11
INE	**73.46 ± 0.20**	**79.26 ± 0.22**	**82.17 ± 0.03**	**93.29 ± 0.05**	**95.49 ± 0.06**	**96.54 ± 0.05**
SWA [21]	74.27 ± 0.12	80.35 ± 0.16	82.15 ± 0.27	93.64 ± 0.18	95.83 ± 0.03	**96.79 ± 0.05**
INE+	**74.39 ± 0.29**	**80.55 ± 0.23**	**82.37 ± 0.04**	**93.71 ± 0.15**	**95.84 ± 0.06**	96.65 ± 0.08
FGE [10]	74.26	79.84	82.27	93.52	95.45	96.36
INE++	**74.65 ± 0.18**	**80.81 ± 0.26**	**82.44 ± 0.10**	**93.73 ± 0.15**	**95.89 ± 0.05**	**96.66 ± 0.12**

VGG: VGG-16, ResNet: ResNet-164, WResNet: WResNet-28-10.

INE+ algorithm, the parameter size is still equivalent to the SGD and **INE** implementation, blessed by the parameter averaging mechanisms.

- **INE++:** This strategy used the same strategy with **INE+** to pre-train and fine-tune the model, while saving a duplication of the intermediate weights pre-trained by SWA. Through averaging the prediction results of **INE+** with the prediction results of SWA, this strategy performed as an ensemble classifier fusing both parameter averaging and prediction averaging mechanisms. Thus, **INE++** had $2K$ parameters.

Again, for a fair comparison, we primarily focused on the comparison of **INE** vs. SGD and **INE+** vs. SWA. Taking its advantage in parameter size, we further compared **INE++** and FGE.

4.1.2 Computational Budgets. As was mentioned before, we fixed the total number of iterations or epochs used in training procedure as the computational budget, since each epoch consumes the same number of gradient estimation computation (i.e., same gradient complexity). Borrowing experiences from Reference [21], we budgeted the computational budget for VGG-16 and WResNet-28-10 as 300 epochs in all experiments, while the budget for ResNet-164 training was set to 225 epochs.

4.2 Comparison under Supervised Learning Settings

Here, we report the performance comparison of **INE** and baseline algorithms. Table 3 presents the performance comparison of supervised learning tasks, where we use performance results of SGD, SWA, and FGE reported in the most recent work [10, 21]. Specifically, we use α of 0.9, 0.8, and 0.8 for PreResNet-164, VGG-16, and WideResNet-28-10, respectively, according to the computational budgets on these architectures. All these models are trained and compared with the same budget settings.

- **Comparisons on k-Parameters Models.** Among all k-parameters models, **INE+** achieves the best performance in the most cases, while SWA and **INE** deliver comparable results.
- **Comparisons on kM-Parameters Models.** Between the two kM-parameters models, **INE++** always outperforms FGE, though FGE uses $3 \sim 6\times$ more parameters than **INE++**.

In summary, we observe a significant performance improvement caused by **INE** and its potential to be incorporated with the state-of-the-art algorithms. As a reference, to understand the power of Teacher-Student training, we also evaluate **INE** without Teacher-Student training (knowledge distillation), where the accuracy of **INE** on VGG-16 is reduced to 72.16 ± 0.19 for CIFAR 100 and 92.89 ± 0.04 for CIFAR 10.

We also compare the total training time for these three architectures on CIFAR-100. As shown in Table 4, the training time does not increase significantly by imposing **INE**, i.e., the additional cost is less than 6% compared with vanilla SGD, even when combined with SWA. However, traditional

Table 4. Training Time (in Seconds) Comparison on CIFAR-100

	VGG	ResNet	WResNet
SGD	7,390	15,116	22,615
SWA	7,497(+1.4%)	15,405(+1.9%)	22,981(+1.6%)
INE	7,458(+0.9%)	15,744(+4.1%)	23,701(+4.8%)
INE+	7,526(+1.8%)	15,909(+5.2%)	23,924(+5.8%)

We perform all experiments on NVIDIA Tesla V100 GPUs and Pytorch1.1. VGG: VGG-16, ResNet: ResNet-164, WResNet: WResNet-28-10.

ensemble learning takes multiple times as training a single model, directly proportional to the number of ensembled models. Therefore, **INE** is efficient in training. Besides, **INE** also reduces the runtime cost, as it only requires a one-shot forward calculation. This advantage is crucial in online services. Compared with SWA, there is a slight increase in training time, because the teacher-student learning operation in **INE** is more time-consuming than weight averaging in SWA.

4.3 Comparison under Transfer Learning Settings

In this section, we report the performance comparison of **INE** and baseline algorithms under transfer learning settings.

4.3.1 Datasets and Algorithm Setups. Several popular transfer learning datasets are used for validation.

- **FGVC-Aircraft** is a challenging benchmark [37] that contains 100 different aircraft model variants with 100 examples per category. They are divided into three equally sized training, validation, and test subsets. Training and test subset are used for the experiment, since we do not perform cross-validation to select hyperparameters.
- **Oxford Flowers 102** [40] consists of 102 flower categories. The chosen flowers are commonly occurring in the United Kingdom; 1020 images are used for training, about only 10 per category, and 6,149 are used for testing.
- **Describable Texture Dataset (DTD)** [4] is a texture database, containing 47 textural categories such as bubbly, cracked, marbled. As studied in Reference [50], textural images usually have quite different high-level representations comparing to images of natural objects. Following Reference [4], we use the training and test subsets with both 1,880 examples for our experiments.
- **Stanford Cars** [24] contains 16,185 images of 196 classes of cars, typically at the level of Make, Model, Year, e.g., 2012 Tesla Model S or 2012 BMW M3 coupe. The data is split into 8,144 training images and 8,041 testing images.

To evaluate the performance in transfer learning settings, we compare **INE** with the state-of-the-art algorithm L^2-SP most recently appeared [31]. For a fair comparison, we compare L^2-SP with **INE** (L^2-SP), which replaces $L(\theta)$ in Equation (5) using the loss function of L^2-SP (i.e., the empirical loss+squared Euclidean distance of the weights of convolutional layers between Teacher and Student networks).

4.3.2 Overall Performance and Comparisons. Table 5 presents the performance comparison of transfer learning tasks, where we compare the performance of **INE** with L^2 and L^2-SP in the most recent work [31]. All algorithms are evaluated using the same settings. To enable transfer learning, all algorithms are based on a pre-trained model from ImageNet. From the table, we observe the

Table 5. Performance on Transfer Learning

	DTD	Aircraft	Flower 102	Cars
ResNet-101				
L^2	0.6553	0.7303	0.8941	0.8903
SWA	0.6677	0.7349	0.9001	0.8889
L^2-SP	0.6915	0.7417	0.8997	0.8955
INE (L^2-SP)	**0.7117**	**0.7777**	**0.9200**	**0.906**
Inception_v3				
L^2	0.6862	0.7987	0.8816	0.8974
SWA	0.6895	0.8026	0.8880	0.8973
L^2-SP	0.7043	0.8176	0.8834	0.8935
INE (L^2-SP)	**0.7128**	**0.8244**	**0.9185**	**0.9129**

significant performance improvement using **INE**, compared to the two baselines. Especially for DTD cases, **INE** (L^2-SP) achieves around 6% accuracy boosts on top of the state-of-the-art algorithms. Blessed by the power of ensemble learning with individual classifiers, **INE** (L^2-SP) outperforms L^2 and L^2-SP in **all** of the evaluated cases.

4.4 Case Studies

To further investigate the source of the performance gain achieved by **INE**, we performed several case studies. All in all, three questions received major concerns as follows:

4.4.1 **Q1. Does diversity between individual classifiers really exist in INE, and if yes, does the diversity improve the accuracy?** One of key assumptions of our research is that the diversity between the individual classifiers could enjoy significant accuracy improvement through ensemble learning. However, all individual classifiers are trained as Student Networks with same Teacher network using the same regularization term. It is reasonable to suspect that every individual classifier converge to the similar points after epochs of iterations.

Q1.1. Though the using random starting points and diversified Teacher-Student training options could bring certain diversity into the learning procedure, we still need direct evidence to support the existence of diversity. *Q1.2* Further, we would like to obtain direct evidence supporting the statement that diversity (as defined in our approach) improves accuracy.

Answer to Q1.1. To characterize the diversity, we measured the **Kullback-Leibler (KL)** divergence divergence between the Logit outputs of every individual classifier. Figure 2 illustrates the averaged KL divergence between every two individual classifiers and KL divergence between every individual classifier and the teacher networks. The results were obtained from the experimental traces based on VGG-16 network and CIFAR-100 datasets with diversified Teacher-Student training options. From the figures, we observe that both divergence are with the decreasing trends (with asymptotic convergence) versus the number of epochs, while both divergences did not converge to zero or small values (the averaged divergence between Teacher and every individual classifier/student converges a number around 1.0, while the divergence between individual classifiers/students reaches to 0.4.). It indicates the obvious diversity exists between the Logit outputs of every individual classifier (i.e., between students) and the one between Teacher and individual classifiers (i.e., between Teacher and every student). In other words, every individual classifier behaves differently, though they are trained using the same Teacher network and the same training datasets.

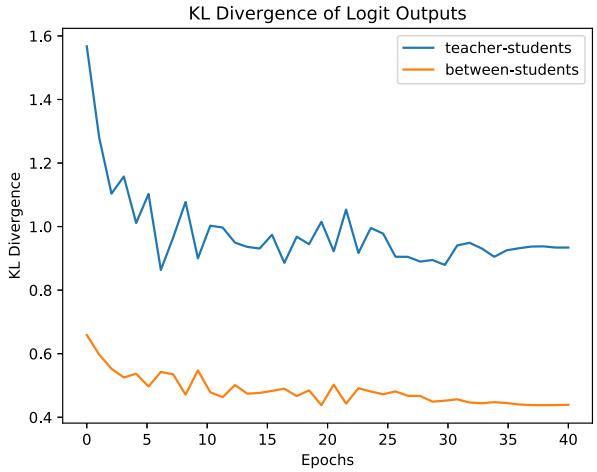

Fig. 2. KL divergence of Logit outputs between individual classifiers (entitled as "between-students"), and between the individual classifiers and the Teacher network (entitled as "teacher-students").

Note that one can prove the existence of diversity through directly measuring the divergence between the weights learned [30]. However, the cases in deep learning are more complicated. Neural networks with different weights can still behave as the same due to the over-parameterization. In this way, our experiments measured the divergences of outputs rather than the weights.

Answer to Q1.2. To validate the contribution of diversity between the crowds of individual classifiers, we designed a new set of experiments. We divided a VGG16 network into $M = 1, 2, 4, \ldots, 16$ individual classifiers for CIFAR-100 datasets training. We specifically focus on the comparison between the testing accuracy of **INE**+ as an ensemble classifier versus the averaged testing accuracy of every individual classifier in **INE**+. Figure 3 demonstrates the comparison results. Obviously, **INE**+ outperforms its individual classifiers (on average) with significant accuracy improvement from 0.1% to 1.7% in Figure 3.

Note that the best performance of **INE**+ (74.39%) can be achieved by $M = 2$, while such improvement made by diversity (i.e., the gap of accuracy) still increases with M the number of individual classifiers divided from the VGG-16. More specifically, we also tested the performance of **INE**+ with $M = 32$ (totally with 32 individual classifiers), where the testing accuracy was 73.18% and the averaged accuracy of individual classifiers fell to 68.31%. The gap was around 4%, though overall testing accuracy is not high due to the poor individual classifiers. It is clear that with the number of increasing M, the number of parameters used by every individual classifier becomes less and less while the performance of every individual classifier would be degraded. It is our great honor to re-investigate the insight of statistical ensemble learning—using a large number of (relatively) weak learners with diversity through ensemble learning can achieve better performance [26]—in deep learning settings, which is different and new. Further, every individual classifier that **INE** ensembles is highly accurate: All of them achieved decent accuracy on CFIAR-100 datasets. It further shows that even strong models (such as well-trained CNNs) are given, certain performance improvement can still be made by **INE** through incorporating diversity and ensemble learning.

4.4.2 **Q2. How does INE improve the performance when the two methods INE+ are combined?** Yet another major concern on the experiment results listed in Table 3 is that **INE**

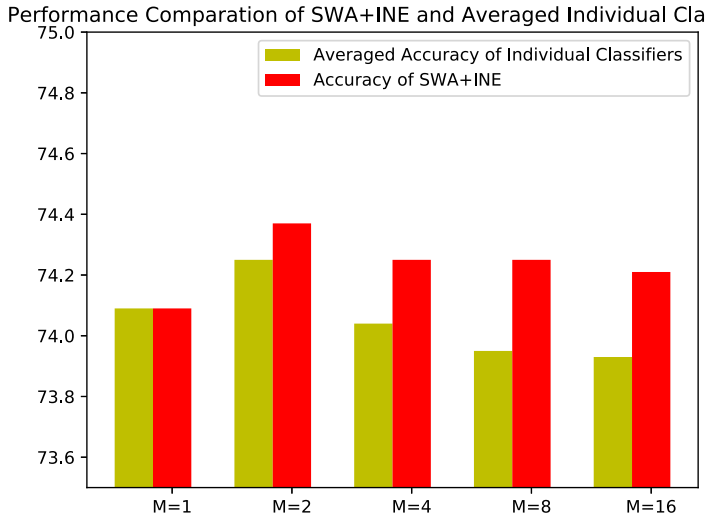

Fig. 3. Comparison between testing accuracy of **INE** versus averaged testing accuracy of every individual classifier with varying M.

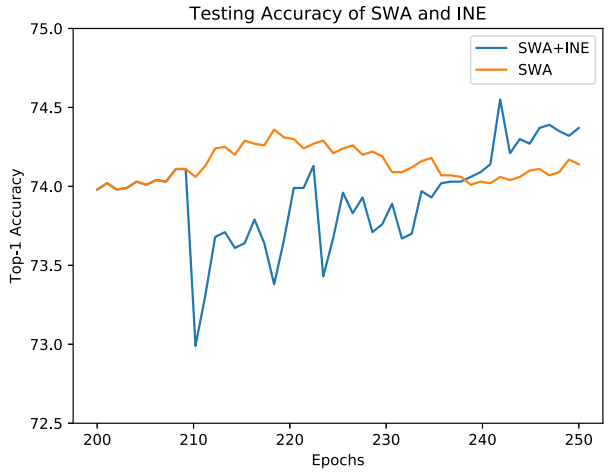

Fig. 4. Comparison between testing accuracy of SWA versus **INE**+ over number of epochs.

standalone cannot always outperform SWA in all cases, while the combined methods **INE**+ could bring significant accuracy boosts in the same computational budget constraints. We are wondering in which way **INE** improves SWA. Specifically, it is our goal in this section to breakdown the overall performance improvement made by **INE**+, i.e., to attribute the performance gain to each part.

Answer to Q2. To answer this question, we measure the testing accuracy of models obtained by SWA and **INE**+ during every epoch of the training procedure. Figure 4 demonstrates the testing accuracy of the two models from the 200th to 250th epochs, when using CIFAR100 datasets and VGG16 network. The two models delivered the same testing accuracy before the 210th epoch, as both of them were trained using SWA before such time-point. Then, **INE**+ suffered a significant

accuracy reduction, since the training algorithm switches to **INE** for training where certain randomization is given to the weights of fully connected layers to initialize individual classifiers (for diversity pursue). Later the testing accuracy of **INE**+ has been quickly recovered with **INE** (diversified Teacher-Student training) algorithm. Finally, **INE**+ outperforms SWA in the budgeted computational time (i.e., the same number of epochs in training procedure) with significant performance improvement.

Note that **INE**+ outperforms SWA at the end of training (i.e., the 250th epoch of the training procedure), while the peak testing accuracy of **INE**+(74.55% achieved at the 242th epoch) is still much higher than the one obtained by SWA (74.36% achieved at the 219th epoch). The performance of SWA would be unstable after the 220th epoch with potential accuracy loss, while **INE** can further improve the accuracy.

4.4.3 **Q3. Can INE ensure the semantic consistency between the individual classifiers and the initial classifier?** As each individual classifier only utilizies a subset of features, it implies that these weak learners may degenerate to the role of attribute recognition that causes the inconsistency between the teacher and students in knowledge distillation.

Answer to Q3. Intuitively, the semantic consistency between the weak learner and the initial task can be almost ensured through "implicitly" incorporating the redundancy in DNN's representations. To test our hypothesis, we adopt the method of visualizing the feature maps [50] of input images based on DNN models, and the results demonstrate that the feature maps extracted from the layers close to the DNN output are capable of representing very high-level semantic information such as a complete object or a part of the animal. The semantic of these visual concepts is usually much broader than an attribute. The fact suggests that features extracted by DNN are much more discriminative than original attributes. In this way, classifiers given a subset of such features as input actually play almost the same role as the initial classification task.

In addition to above explanation, we provide extensive empirical studies with supporting evidences. First, we have shown in Figure 3 that the accuracy of each individual classifier is marginally lower than the initial classifier (corresponding to the case with M = 1) as the number of individual classifiers increases. Second, we demonstrate the behaviors of each weak classifier by visualizing the activation maps of the last representation layer that is closest to the output in Figure 5. We employ INE with eight individual classifiers to train ResNet-101 on Stanford Cars. Results are compared with the initial classification task for clear demonstration. From the images in Figure 5, we can observe that the initial task (the left-most column) utilizes most of important attributes to recognize a car, while weak classifiers have different weights on these attributes, for example, the classifier #4 usually pays more attention to car windows than other weak classifiers. These examples provide the direct measurement and evidence to the diversity between classifiers.

5 RELATED WORKS AND DISCUSSION

Ensemble learning [6] is a widely used learning technique, where multiple base classifiers are first trained and then aggregated to improve learning accuracy [26]. In traditional statistical learning settings, the base classifiers trained with the convex losses could be improved with lower variance and bounded bias through simple model averaging [9]. Recent studies [8, 10, 19, 21, 27] demonstrated amazing insights that model averaging still works in deep ensemble learning, where the loss is believed to be strongly nonconvex.

In this section, we categorize and review the current deep ensemble learning methods in three folders—*implicit ensemble learning*, *explicit ensemble learning*, and *auto-ensemble learning* for deep neural networks.

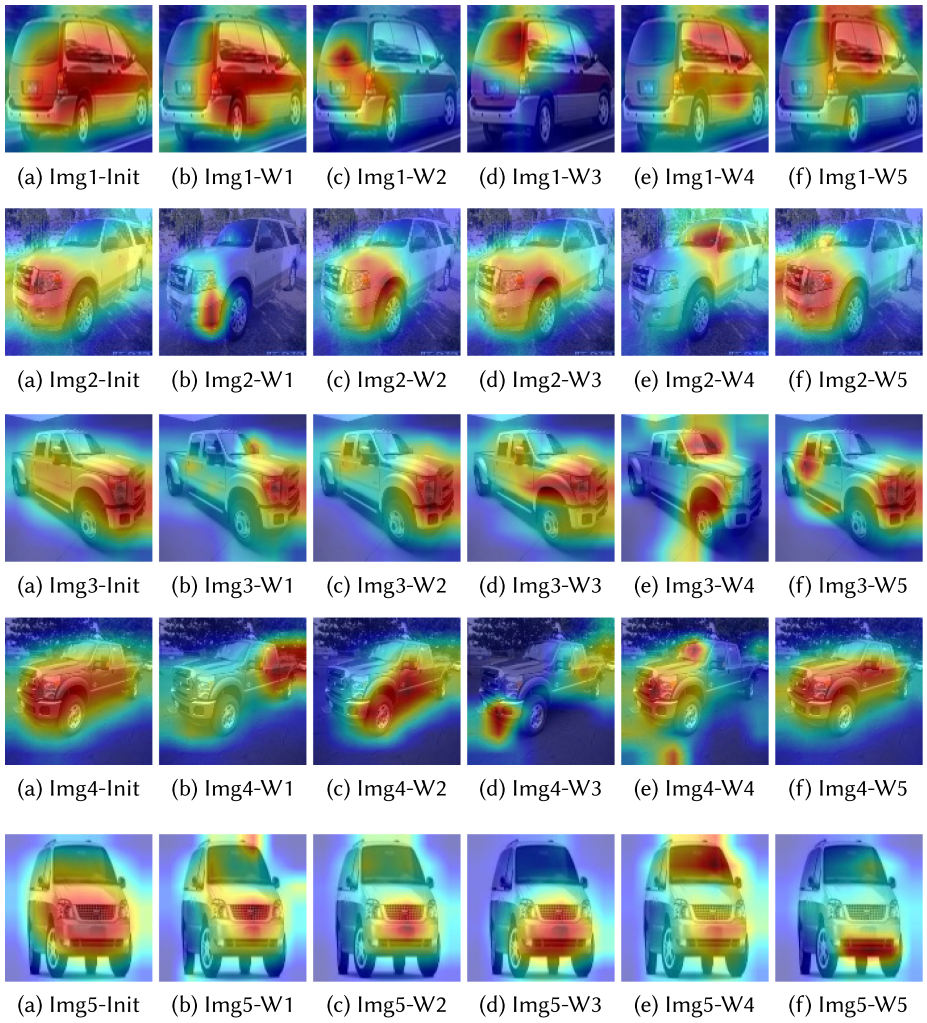

Fig. 5. Comparisons on the behaviors of the initial task and weak classifiers by visualizing their activation maps of the last representation layer. The model is trained on ResNet-101 over the Stanford Cars dataset. Due to space limitation, we only show the activation maps of the first five among the entire collection of eight weak classifiers.

5.1 Implicit Ensemble Learning for Deep Neural Networks

There is a group of stochastic regularization techniques that can be viewed as discovering base models and performing model averaging implicitly. For example Dropout [18] works by randomly selecting a subset of hidden units in the learning process and averages over the units during inference time. It can be viewed as an economical approximation to train a very large ensemble of neural networks [2, 16]. Swapout [43] extends the idea of dropout and samples from a richer set of architectures, which includes stochastic depth and residual architectures as special cases. Shakeout [22] chooses to enhance or reverse the contribution of a hidden unit, instead of simply dropping it. Branchout [14] utilizes the multi-branch structure in convolutional networks and randomly drops

an entire path. Other techniques falling in this category include dropconnect [46], bridgeout [23], and blockout [39].

5.2 Explicit Ensemble Learning for Deep Neural Networks

Different from implicitly aggregating a group of base models, recently there is a new thread of research for first explicitly identifying such models and then aggregating them [15, 25, 41]. For example, Huang et al. [19] propose snapshot ensembles, a framework of training a single neural network, taking snapshots along its optimization path, and hence obtaining an ensemble of networks at NO additional cost of training one model. Snapshot adopts *Cyclic Cosine Annealing* strategy [36] to warmly restart the training procedure periodically and picks up the snapshot per the training procedure converging (to some local minimums). Extending the idea of snapshot ensembling, Laine & Aila proposed *"temporal ensembling,"* where similar "snapshots" models were obtained from the "network-in-training" and the final results are given by the majority voting of the prediction results based on the snapshots [27]. To encourage diversity among snapshots, the authors adopted various data augmentation and training settings per epoch. Moreover, Reference [8] extended the temporal ensemble strategy [27] with a *"mean teacher"* [45] for ensemble transfer learning under domain adaption settings. Xie et al. [48] propose horizontal voting, vertical voting, and horizontal stacked ensemble methods to improve the classification accuracy. Mosca and Magoulas [38] propose deep incremental boosting that uses internal knowledge of convolutional nets to generate ensembles quickly.

5.3 Auto-ensemble Learning for Deep Neural Networks

Auto-ensemble utilizes **neural architecture search (NAS)** for automated base model discovery. For example, in contrast to aggregating networks of predetermined architectures, AdaNet adaptively learns the network structure and weights at the same time [5]. It starts from a simple linear model and gradually adds hidden units and layers based on theoretical estimates of generalization errors. It produces competitive performance on several binary classification tasks taken from CIFAR-10 dataset. Due to the high computational cost of NAS, auto-ensemble has not been widely used yet.

5.4 Discussion on the Most Relevant Works

Our work falls into the category of explicitly deep ensemble learning. The work that is highly related to what we are doing is ***Stochastic Weight Averaging*** **(SWA)** [21]. The authors have found that the geometry of local minima provides key insights for model ensemble. SWA traverses the landscape of the loss function and averages the weights of multiple snapshots via a **Bayesian parameter averaging (BPA)** method to obtain the final model. Comparing SWA to other explicit ensemble algorithms [10, 19, 43**?**], the advantages of SWA are significant: (1) SWA employs a smaller number of parameters for inference. Suppose M snapshots were taken from the training procedure, the number of parameters used in SWA is equal to that of a single snapshot while the number of parameters used in the many other methods is M fold larger than SWA. (2) SWA enjoys the state-of-the-art of classification accuracy with a small number of parameters.

Comparing to existing methods [10, 19, 21, 43**?**], our method is novel on several counts. First, we discover base models by splitting one network into multiple subnetworks, rather than considering the training trajectory. Second, we encourage the diversity of those networks through a novel knowledge distillation with a method that we call diversified Teacher-Student training. Empirically, our proposed method **INE** achieves the state-of-the-art performance and delivers better performance than competing method such as SWA on a number of real-world applications. In our future work, we plan to improve the performance of **INE** using advanced knowledge distillation

regularizers [33, 47], network architectures with multiple parallel sub-networks [13, 51], and noisy training procedures [20, 32, 44] to further strengthen diversity in models.

6 CONCLUSION

In this article, we proposed a new ensemble deep learning method that discovers its base classifier explicitly with a budget on computational resources using a small size of parameters. We do so by first subdividing the classifier of a deep learning model into multiple parts and then using a diversified knowledge distillation method to train each of them to obtain highly accurate yet diverse base classifiers. Our extensive experimental study using both supervised learning and transfer learning has confirmed the advantage of the proposed method on a wide range of popular network architectures (e.g., ResNet, VGG, and Wide ResNet), benchmark datasets (e.g., ImageNet, CIFAR-10, CIFAR-100, DTD), and learning tasks.

In our experiments, the proposed method **INE** can achieve the best of its accuracy when using the teacher network trained by SWA [21]. We believe the outstanding performance is in part due to the aggregation of diversity pursued through the so-called stochastic weight averaging scheme. It thus makes sense for us to incorporate more ensemble trainers to diversify the snapshots obtained and enhance the overall performance. In the future, we plan to further improve the diversity-encouraging training and consider training multiple models simultaneously.

ACKNOWLEDGEMENTS

Authors would like to thank reviewers' efforts in evaluating this manuscript and editors' efforts in organizing the reviewing process. Please feel free to download the source code of our algorithms and reproduce our experiments using the link https://github.com/INECode/In-Network-Ensemble.

REFERENCES

[1] Zeyuan Allen-Zhu, Yuanzhi Li, and Zhao Song. 2019. A convergence theory for deep learning via over-parameterization. In *International Conference on Machine Learning*. PMLR, 242–252.

[2] Pierre Baldi and Peter J. Sadowski. 2013. Understanding dropout. In *Advances in Neural Information Processing Systems*. 2814–2822.

[3] Gérard Biau, Luc Devroye, and Gábor Lugosi. 2008. Consistency of random forests and other averaging classifiers. *J. Mach. Learn. Res.* 9, Sept. (2008), 2015–2033.

[4] M. Cimpoi, S. Maji, I. Kokkinos, S. Mohamed, , and A. Vedaldi. 2014. Describing textures in the wild. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[5] Corinna Cortes, Xavier Gonzalvo, Vitaly Kuznetsov, Mehryar Mohri, and Scott Yang. 2017. Adanet: Adaptive structural learning of artificial neural networks. In *34th International Conference on Machine Learning*. JMLR.org, 874–883.

[6] Thomas G. Dietterich. 2000. Ensemble methods in machine learning. In *International Workshop on Multiple Classifier Systems*. Springer, 1–15.

[7] Jonathan Frankle and Michael Carbin. 2018. The lottery ticket hypothesis: Finding sparse, trainable neural networks. arXiv preprint arXiv:1803.03635 (2018).

[8] Geoff French, Michal Mackiewicz, and Mark Fisher. 2018. Self-ensembling for visual domain adaptation. In *International Conference on Learning Representations*.

[9] Yoav Freund, Yishay Mansour, Robert E. Schapire et al. 2004. Generalization bounds for averaged classifiers. *Ann. Statist.* 32, 4 (2004), 1698–1722.

[10] Timur Garipov, Pavel Izmailov, Dmitrii Podoprikhin, Dmitry P. Vetrov, and Andrew G. Wilson. 2018. Loss surfaces, mode connectivity, and fast ensembling of DNNs. In *International Conference on Advances in Neural Information Processing Systems 31*, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett (Eds.). Curran Associates, Inc., 8803–8812.

[11] Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *13th International Conference on Artificial Intelligence and Statistics*. 249–256.

[12] Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. 2016. *Deep Learning*. Vol. 1. The MIT Press, Cambridge, MA.

[13] Ian Goodfellow, David Warde-Farley, Mehdi Mirza, Aaron Courville, and Yoshua Bengio. 2013. Maxout networks. In *International Conference on Machine Learning*. PMLR, 1319–1327.
[14] Bohyung Han, Jack Sim, and Hartwig Adam. 2017. Branchout: Regularization for online ensemble tracking with convolutional neural networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 521–530.
[15] Lars Kai Hansen and Peter Salamon. 1990. Neural network ensembles. *IEEE Trans. Pattern Anal. Mach. Intell.* 12, 10 (1990), 993–1001.
[16] Kazuyuki Hara, Daisuke Saitoh, and Hayaru Shouno. 2016. Analysis of dropout learning regarded as ensemble learning. In *International Conference on Artificial Neural Networks*. Springer, 72–79.
[17] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the knowledge in a neural network. arXiv preprint arXiv:1503.02531 (2015).
[18] Geoffrey E. Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R. Salakhutdinov. 2012. Improving neural networks by preventing co-adaptation of feature detectors. arXiv preprint arXiv:1207.0580 (2012).
[19] Gao Huang, Yixuan Li, Geoff Pleiss, Zhuang Liu, John E. Hopcroft, and Kilian Q. Weinberger. 2017. Snapshot ensembles: Train 1, get M for free. In *International Conference on Learning Representations*.
[20] Gao Huang, Yu Sun, Zhuang Liu, Daniel Sedra, and Kilian Q. Weinberger. 2016. Deep networks with stochastic depth. In *European Conference on Computer Vision*. Springer, 646–661.
[21] Pavel Izmailov, Dmitrii Podoprikhin, Timur Garipov, Dmitry Vetrov, and Andrew Gordon Wilson. 2018. Averaging weights leads to wider optima and better generalization. In *Conference on Uncertainty in Artificial Intelligence (UAI)*.
[22] Guoliang Kang, Jun Li, and Dacheng Tao. 2018. Shakeout: A new approach to regularized deep neural network training. *IEEE Trans. Pattern Anal. Mach. Intell.* 40, 5 (2018), 1245–1258.
[23] Najeeb Khan, Jawad Shah, and Ian Stavness. 2018. Bridgeout: Stochastic bridge regularization for deep neural networks. arXiv preprint arXiv:1804.08042 (2018).
[24] Jonathan Krause, Michael Stark, Jia Deng, and Li Fei-Fei. 2013. 3D object representations for fine-grained categorization. In *4th International IEEE Workshop on 3D Representation and Recognition (3dRR-13)*.
[25] Anders Krogh and Jesper Vedelsby. 1995. Neural network ensembles, cross validation, and active learning. In *International Conference on Advances in Neural Information Processing Systems*. 231–238.
[26] Ludmila I. Kuncheva and Christopher J. Whitaker. 2003. Measures of diversity in classifier ensembles and their relationship with the ensemble accuracy. *Machine Learn.* 51, 2 (2003), 181–207.
[27] Samuli Laine and Timo Aila. 2017. Temporal ensembling for semi-supervised learning. In *International Conference on Learning Representations*.
[28] Quoc V. Le, Jiquan Ngiam, Adam Coates, Abhik Lahiri, Bobby Prochnow, and Andrew Y. Ng. 2011. On optimization methods for deep learning. In *28th International Conference on International Conference on Machine Learning*. Omnipress, 265–272.
[29] Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf. 2016. Pruning filters for efficient convnets. arXiv preprint arXiv:1608.08710 (2016).
[30] Nan Li, Yang Yu, and Zhi-Hua Zhou. 2012. Diversity regularized ensemble pruning. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 330–345.
[31] Xuhong Li, Yves Grandvalet, and Franck Davoine. 2018. Explicit inductive bias for transfer learning with convolutional networks. In *35th International Conference on Machine Learning*.
[32] Xingjian Li, Haoyi Xiong, Haozhe An, Cheng-Zhong Xu, and Dejing Dou. 2020. RIFLE: Backpropagation in depth for deep transfer learning through Re-Initializing the Fully-connected LayEr. In *International Conference on Machine Learning*. PMLR, 6010–6019.
[33] Xingjian Li, Haoyi Xiong, Hanchao Wang, Yuxuan Rao, Liping Liu, and Jun Huan. 2019. Delta: Deep learning transfer using feature map with attention for convolutional networks. In *International Conference on Learning Representations*.
[34] Yuanzhi Li and Yingyu Liang. 2018. Learning overparameterized neural networks via stochastic gradient descent on structured data. *Adv. Neural Inf. Process. Syst.* 31 (2018), 8157–8166.
[35] Zhizhong Li and Derek Hoiem. 2017. Learning without forgetting. *IEEE Trans. Pattern Anal. Mach. Intell.* 40, 12 (2017), 2935–2947.
[36] Ilya Loshchilov and Frank Hutter. 2016. SGDR: Stochastic gradient descent with warm restarts. In *International Conference on Learning Representations*.
[37] S. Maji, J. Kannala, E. Rahtu, M. Blaschko, and A. Vedaldi. 2013. *Fine-grained Visual Classification of Aircraft*. arXiv preprint arXiv:1306.5151.
[38] Alan Mosca and George D. Magoulas. 2017. Deep incremental boosting. arXiv preprint arXiv:1708.03704 (2017).
[39] Calvin Murdock, Zhen Li, Howard Zhou, and Tom Duerig. 2016. Blockout: Dynamic model selection for hierarchical deep networks. In *IEEE Conference on Computer Vision and Pattern Recognition*. 2583–2591.
[40] M.-E. Nilsback and A. Zisserman. 2008. Automated flower classification over a large number of classes. In *Indian Conference on Computer Vision, Graphics and Image Processing*.

[41] Michael P. Perrone and Leon N. Cooper. 1992. *When Networks Disagree: Ensemble Methods for Hybrid Neural Networks*. Technical Report. Institute for Brain and Neural Systems, Brown University, Providence, RI.
[42] Matthias Seeger, John Langford, and Nimrod Megiddo. 2001. An improved predictive accuracy bound for averaging classifiers. In *18th International Conference on Machine Learning*. 290–297.
[43] Saurabh Singh, Derek Hoiem, and David Forsyth. 2016. Swapout: Learning an ensemble of deep architectures. In *International Conference on Advances in Neural Information Processing Systems*. 28–36.
[44] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.* 15, 1 (2014), 1929–1958.
[45] Antti Tarvainen and Harri Valpola. 2017. Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. In *International Conference on Advances in Neural Information Processing Systems*. 1195–1204.
[46] Li Wan, Matthew Zeiler, Sixin Zhang, Yann Le Cun, and Rob Fergus. 2013. Regularization of neural networks using dropconnect. In *International Conference on Machine Learning*. 1058–1066.
[47] Ruosi Wan, Haoyi Xiong, Xingjian Li, Zhanxing Zhu, and Jun Huan. 2019. Towards making deep transfer learning never hurt. In *IEEE International Conference on Data Mining (ICDM)*. IEEE, 578–587.
[48] Jingjing Xie, Bing Xu, and Zhang Chuang. 2013. Horizontal and vertical ensemble with deep representation for classification. arXiv preprint arXiv:1306.2759 (2013).
[49] Junho Yim, Donggyu Joo, Jihoon Bae, and Junmo Kim. 2017. A gift from knowledge distillation: Fast optimization, network minimization and transfer learning. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.
[50] Matthew D. Zeiler and Rob Fergus. 2014. Visualizing and understanding convolutional networks. In *European Conference on Computer Vision*. Springer, 818–833.
[51] Baoxin Zhao, Haoyi Xiong, Jiang Bian, Zhishan Guo, Cheng-Zhong Xu, and Dejing Dou. 2021. COMO: Widening deep neural networks with COnvolutional MaxOut. *IEEE Trans. Multimedia* 23 (2021), 1722–1730.

Received January 2021; revised June 2021; accepted June 2021