Adversarial-Based Knowledge Distillation for Multi-Model Ensemble and Noisy Data Refinement

Zhiqiang Shen, Zhankui He, Wanyun Cui, Jiahui Yu, Yutong Zheng, Chenchen Zhu, Marios Savvides

Abstract—Generic Image recognition is a fundamental and fairly important visual problem in computer vision. Recently with the development of modern neural network design and automatic architecture search, this task has been one of the fastest moving areas and achieved surprising results which even surpasses human-level performance on large-scale datasets like ImageNet [1] and OpenImage [2]. One of the major challenges of this task lies in the fact that single image usually has multiple objects inside while the labels are still one-hot, another one is noisy and sometimes missing labels when annotated by humans. In this paper, we focus on tackling these challenges accompanying with two different image recognition problems: multi-model ensemble and noisy data recognition with a unified framework, showing the superiority and advantages of our proposed method. As is well-known, usually the best performing deep neural models are ensembles of multiple base-level networks, as it can mitigate the variation or noise containing in the dataset. Unfortunately, the space required to store these many networks, and the time required to execute them at runtime, prohibit their use in applications where test sets are large (e.g., ImageNet). In this paper, we present a method for compressing large, complex trained ensembles into a single network, where the knowledge from a variety of trained deep neural networks (DNNs) is distilled and transferred to a single DNN. In order to distill diverse knowledge from different trained (teacher) models, we propose to use adversarial-based learning strategy where we define a block-wise training loss to guide and optimize the predefined student network to recover the knowledge in teacher models, and to promote the discriminator network to distinguish teacher vs. student features simultaneously. The proposed ensemble method (MEAL) of transferring distilled knowledge with adversarial learning exhibits three important advantages: (1) the student network that learns the distilled knowledge with discriminators is superiorly optimized than the original model; (2) fast inference is realized by a single forward pass, while the performance is even better than traditional ensembles from multi-original models; (3) the soft distributions from teacher networks can provide more informative and accurate supervision signals in the image for student learning, which can overcome the multi-object and noisy label problem to some extent. Extensive experiments on CIFAR-10/100, SVHN, ImageNet and iMaterialist Challenge Dataset demonstrate the effectiveness of our MEAL method. On ImageNet, our ResNet-50 based MEAL achieves top-1/5 21.79%/5.99% val error, which outperforms the original model by 2.06%/1.14%. On iMaterialist Challenge Dataset, our MEAL obtains a remarkable improvement of top-3 1.15% (official evaluation metric) on a strong baseline model of ResNet-101.

Index Terms—Adversarial Learning, Knowledge Distillation, Multi-Model Ensemble, Noisy Data Refinement.

INTRODUCTION

The model ensemble approach is a collection of neural networks whose predictions are combined at test stage by weighted averaging or voting. It has been long observed that ensembles of multiple networks are generally much more robust and accurate than a single network if the training data is noisy and intractable to handle. This benefit has also been exploited indirectly when training a single network through Dropout [3], Dropconnect [4], Stochastic Depth [5], Swapout [6], etc. We extend this idea by forming ensemble predictions during training, using the outputs of different network architectures with different or identical augmented input. Our testing still operates on a single network, but the supervision labels made on different pretrained networks correspond to an ensemble prediction of a group of individual reference networks.

The traditional ensemble, or called true ensemble, has some disadvantages that are often overlooked. 1) Redundancy: The information or knowledge contained in the trained neural networks are always redundant and has overlaps between with each other. Directly combining the predictions often requires extra computational cost but the gain is limited. 2) Ensemble is always large and slow: Ensemble requires more computing operations than an individual network, which makes it unusable for applications with limited memory, storage space, or computational power such as desktop, mobile and even embedded devices, and for applications in which real-time predictions are needed.

To address the aforementioned shortcomings, in this paper we propose to use a learning-based ensemble method. Our goal is to learn an ensemble of multiple neural networks...
without incurring any additional testing costs, as shown in Fig. 2. We achieve this goal by leveraging the combination of diverse outputs from different neural networks as supervisions to guide the target network training. The reference networks are called Teachers and the target networks are called Students. Instead of using the traditional one-hot vector labels, we use the soft labels that provide more coverage for co-occurring and visually related objects and scenes. We argue that labels should be informative for the specific image. In other words, the labels should not be identical for all the given images with the same class. More specifically, as shown in Fig. 3, an image of "tobacco shop" has similar appearance to "library" should have a different label distribution than an image of "tobacco shop" but is more similar to "grocery store". It can also be observed that soft labels can provide the additional intra- and inter-category relations of datasets.

To further improve the robustness of student networks, we introduce an adversarial learning strategy to force the student to generate similar outputs as teachers. We propose two different strategies for the generative adversarial training: (i) joint training with a unified framework; and (ii) alternately update gradients with separate training processes, i.e., updating the gradients in discriminator and student network iteratively. To the best of our knowledge, there are very few existing works adopting generative adversarial learning to force the student networks to have similar distribution outputs with the teachers, so our proposed method is a pioneer of this direction for multi-model ensemble. Our experiments show that MEAL consistently improves the accuracy across a variety of popular network architectures and different datasets. For instance, our shake-shake [8] based MEAL achieves 2.54% test error on CIFAR-10, which is a relative 11.2% improvement. On ImageNet, our ResNet-50 based MEAL achieves 21.79%/5.99% val error, which outperforms the baseline by a large margin.

Furthermore, we extend our method to the problem of noisy data processing. We propose an iterative refinement paradigm based on our MEAL method, which can refine the labels from the teacher networks progressively and provide more accurate supervisions for the student network training. We conduct experiments on iMaterialist Challenge Dataset and the results show that our method can vastly improve the performance of base models.

Fig. 1: Visualizations of validation images from the ImageNet dataset [1] by t-SNE [7]. We randomly sample 10 classes within 1000 classes. Left is the single model result using the standard training strategy. Right is our MEAL ensemble model result.

Fig. 2: Comparison of FLOPs at inference time. Huang et al. [10] employ models at different local minimum for ensembling, which enables no additional training cost, but the computational FLOPs at test time linearly increase with more ensembles. In contrast, our method use only one model during inference time throughout, so the testing cost is independent of # ensembles.

In summary, our contribution in this paper is three fold:

- An end-to-end framework with adversarial learning is designed based on the teacher-student learning paradigm for deep neural network ensembling and noisy data learning.
- The proposed method can achieve the goal of ensembling multiple neural networks with no additional testing cost.
- The proposed method improves the state-of-the-art accuracy on CIFAR-10/100, SVHN, ImageNet and iMaterialist Challenge Dataset for a variety of existing network architectures.

A preliminary version of this manuscript [9] has been published in a previous conference. In this version, we involved and compared two different gradient update strategies for adversarial learning on our proposed MEAL framework. We also provided a novel learning paradigm for how to adopt our method on handling noisy data circumstances. Furthermore, we included more experiments, details, analysis and an iterative refinement strategy with better performance. Currently, there are few works focusing on adopting generative adversarial learning on feature space for learning identical distributions between teacher and student networks. Thus, this work gives very good and practical guidelines for multi-model learning/ensemble and noisy data refinement.

2 Related Work

There is a large body of previous work [11], [12], [13], [14], [10], [15], [16], [17] on ensembles with neural networks.
focus on the cost of testing ensembles. In contrast to the Snapshot Ensembles, here we propose to address the cost of training generalization of an individual network. Recently, Snapshot Ensembles [10] is proposed to address the cost of training.

However, most of these prior studies focus on improving the generalization of an individual network. Recently, Snapshot Ensembles [10] is proposed to address the cost of training ensembles. In contrast to the Snapshot Ensembles, here we focus on the cost of testing ensembles. Our method is based on the recently raised knowledge distillation [18], [19], [20], [21] and adversarial learning [22], so we will review the ones that are most directly connected to our work.

“Implicit” Ensembling. Essentially, our method is an “implicit” ensemble which usually has high efficiency during both training and testing. The typical “implicit” ensemble methods include: Dropout [3], DropConnection [4], Stochastic Depth [5], Swapout [6], etc. These methods generally create an exponential number of networks with shared weights during training and then implicitly ensemble them at test time. In contrast, our method focuses on the subtle differences of labels with identical input. Perhaps the most similar to our work is the recent proposed Label Refinery [23], who focus on the single model refinement using the softened labels from the previous trained neural networks and iteratively learn a new and more accurate network. Our method differs from it in that we introduce adversarial modules to force the model to learn the difference between teachers and students, which can improve model generalization and can be used in conjunction with any other implicit ensembling techniques. There are some other ensemble methods like DivE2 [17], which aims to train an ensemble of models that assigns data to models at each training epoch based on each models current expertise and an intra- and inter-model diversity reward. It starts by choosing easy samples for each model, and then gradually adjusts towards the models having specialized and complementary expertise on subsets of the training data.

Adversarial Learning. Generative Adversarial Learning [22] is firstly proposed to generate realistic-looking images from random noise using neural networks. It consists of two components. One serves as a generator and another one as a discriminator. The generator is used to synthesize images to fool the discriminator, meanwhile, the discriminator tries to distinguish real and fake images. Recently, numerous interesting GAN evolution algorithms have been proposed, such as Wasserstein GAN [24], Improved Wasserstein gans [25], DRAGAN [26], NS GAN [27], LS GAN [28], et al. Generally, the generator and discriminator are trained simultaneously through competing with each other. In this work, we employ generators to synthesize student features and use discriminator to discriminate between teacher and student outputs for the same input image. An advantage of adversarial learning is that the generator tries to produce similar features as a teacher that the discriminator cannot differentiate. This procedure improves the robustness of training for student network and has applied to many fields such as image-to-image translation [29], [30], [31], [32], [33], [34], image generation [35], detection [36], etc.

Knowledge Transfer. Distilling knowledge from trained neural networks and transferring it to another new network has been well explored in [18], [37], [20], [38], [21], [39], [23], [40], [41], [42], [43], [44], [45], [46], [47]. The typical way of transferring knowledge is the teacher-student learning paradigm, which uses a softened distribution of the final output of a teacher network to teach information to a student network. With this teaching procedure, the student can learn how a teacher studied given tasks in a more efficient form. Yim et al. [21] defined the distilled knowledge to be transferred flows between different intermediate layers and computed the inner product between parameters from two networks. Bagherinezhad et al. [23] studied the effects of various properties of labels and introduce the Label Refinery method that iteratively updated the ground truth labels after examining the entire dataset with the teacher-student learning paradigm. Park et al. [42] introduced a novel dubbed relational knowledge distillation (RKD) that transferred mutual relations of data examples instead. For concrete realizations of RKD, they proposed distance-wise and angle-wise distillation losses that penalize structural differences in relations.

Learning with Noisy Labels. Learning with noisy labels has been a widely-explored research topic in recent years, since it has wide usage and applications [48]. There are a large number of variety methods which are applied to tackle this problem of modeling distribution of noisy and true annotations, such as knowledge graphs and distillation [20], conditional random fields [49], directed graphical models [50], etc. As in the deep learning era, one line to handle this problem with neural networks is [48], [51], [52], [53], [54], [55], [56], [57], which are formulated by explicit or implicit noisy models.

In particular, Li et al. [48] proposed to use a unified distillation framework to adopt side information, including a small clean dataset and label relations in knowledge graph, to hedge the risk of learning from noisy labels. Liu et al. [58] presented an importance reweighting framework for classification in the presence of label noise which sample labels are randomly corrupted. Li et al. [48] proposed a noise-tolerant learning algorithm, which a meta-learning update strategy is performed prior to conventional gradient update operation. The proposed meta-learning method simulates actual training by generating synthetic noisy labels, and train the model after one gradient update using each set of synthetic noisy labels. Sukhbaatar et al. [54] explored the performance of discriminatively-trained CNN when training on noisy data. They introduced an extra noise layer into the network which adapted the network outputs to match the noisy label distribution. The parameters of this noise layer can be estimated as part of the training.
process and involve simple modifications to current training infrastructures for deep neural networks.

3 OVERVIEW

Siamese-like Network Structure Our framework is a siamese-like architecture that contains two-stream networks in teacher and student branches. The structures of two streams can be identical or different, but should have the same number of blocks, in order to utilize the intermediate outputs. The whole framework of our method is shown in Fig. 4. It consists of a teacher network, a student network, alignment layers, similarity loss layers and discriminators.

The teacher and student networks are processed to generate intermediate outputs for alignment. The alignment layer is an adaptive pooling process that takes the same or different length feature vectors as input and output fixed-length new features. We force the model to output similar features of student and teacher by training student network adversarially against several discriminators. We will elaborate each of these components in the following sections with more details.

4 ADVERSARIAL LEARNING (AL) FOR KNOWLEDGE DISTILLATION

4.1 Similarity Measurement

Given a dataset \( D = (X_i, Y_i) \), we pre-trained the teacher network \( T_\theta \) over the dataset using the cross-entropy loss against the one-hot image-level labels\(^2\) in advance. The student network \( S_\theta \) is trained over the same set of images, but uses labels generated by \( T_\theta \). More formally, we can view this procedure as training \( S_\theta \) on a new labeled dataset \( \tilde{D} = (X_i, T_\theta(X_i)) \). Once the teacher network is trained, we freeze its parameters when training the student network.

We train the student network \( S_\theta \) by minimizing the similarity distance between its output and the soft label generated by the teacher network. Letting \( p_{c}^{T_\theta}(X_i) = T_\theta(X_i)[c] \), \( p_{c}^{S_\theta}(X_i) = S_\theta(X_i)[c] \) be the probabilities assigned to class \( c \) in the teacher model \( T_\theta \) and student model \( S_\theta \). The similarity metric can be formulated as:

\[
\mathcal{L}_{\text{Sim}} = d(T_\theta(X_i), S_\theta(X_i)) = \sum_c d(p_{c}^{T_\theta}(X_i), p_{c}^{S_\theta}(X_i))
\] (1)

We investigated three distance metrics in this work, including \( \ell_1 \), \( \ell_2 \) and KL-divergence. The detailed experimental comparisons are shown in Tab. 1. Here we formulate them as follows.

\( \ell_1 \) distance is used to minimize the absolute differences between the estimated student probability values and the reference teacher probability values. Here we formulate it as:

\[
\mathcal{L}_{\ell_1, \text{Sim}}(S_\theta) = \frac{1}{n} \sum_{c} \sum_{i=1}^{n} \left| p_{c}^{T_\theta}(X_i) - p_{c}^{S_\theta}(X_i) \right|
\] (2)

\( \ell_2 \) distance or euclidean distance is the straight-line distance in euclidean space, which has been used in Mean Teacher [39] (mean squared error, MSE) as the consistency loss. We use \( \ell_2 \) loss function to minimize the error which is the sum of all squared differences between the student output probabilities and the teacher probabilities. The \( \ell_2 \) can be formulated as:

\[
\mathcal{L}_{\ell_2, \text{Sim}}(S_\theta) = \frac{1}{n} \sum_{c} \sum_{i=1}^{n} \left( p_{c}^{T_\theta}(X_i) - p_{c}^{S_\theta}(X_i) \right)^2
\] (3)

KL-divergence is a measure of how one probability distribution is different from another reference probability distribution. Here we train student network \( S_\theta \) by minimizing

\[\ldots\]
the KL-divergence between its output $p^S_{c}(X_i)$ and the soft labels $p^T_{c}(X_i)$ generated by the teacher network. Our loss function is:

$$\mathcal{L}_{KL_{Sim}}(S_\theta) = -\frac{1}{n} \sum_{c} \sum_{i=1}^{n} p^T_{c}(X_i) \log (p^S_{c}(X_i) / p^T_{c}(X_i))$$

$$= -\frac{1}{n} \sum_{c} \sum_{i=1}^{n} p^T_{c}(X_i) \log p^S_{c}(X_i) \quad (4)$$

$$+ \frac{1}{n} \sum_{c} \sum_{i=1}^{n} p^T_{c}(X_i) \log p^T_{c}(X_i)$$

where the second term is the entropy of soft labels from teacher network and is constant with respect to $T_\theta$. We can remove it and simply minimize the cross-entropy loss as follows:

$$\mathcal{L}_{CE_{Sim}}(S_\theta) = -\frac{1}{n} \sum_{c} \sum_{i=1}^{n} p^T_{c}(X_i) \log p^S_{c}(X_i) \quad (5)$$

### 4.2 Intermediate Alignment

**Adaptive Pooling.** The purpose of the adaptive pooling layer is to align the intermediate output from teacher network and student network. This kind of layer is similar to the ordinary pooling layer like average or max pooling, but can generate a predefined length of output with different input size. Because of this specialty, we can use the different teacher networks and pool the output to the same length of student output. Pooling layer can also achieve spatial invariance when reducing the resolution of feature maps. Thus, for the intermediate output, our loss function is:

$$\mathcal{L}_{Sim}^j = d(f(T_\theta), f(S_\theta)) \quad (6)$$

where $T_\theta$ and $S_\theta$ are the outputs at $j$-th layer of the teacher and student, respectively. $f$ is the adaptive pooling function that can be average or max. Fig. 8 illustrates the process of adaptive pooling. Because we adopt multiple intermediate layers, our final similarity loss is a sum of individual one:

$$\mathcal{L}_{Sim} = \sum_{j \in A} \mathcal{L}_{Sim}^j \quad (7)$$

where $A$ is the set of layers that we choose to produce output. In our experiments, we use the last layer in each block of a network (block-wise).

### 4.3 Stacked Discriminators

We generate student output by training the student network $S_\theta$ and freezing the teacher parts adversarially against a series of stacked discriminators $D_j$. A discriminator $D$ attempts to classify its input $x$ as teacher or student by maximizing the following objective as in [22]:

$$\mathcal{L}^j_{GAN} = \mathbb{E}_{x \sim p_{\text{teacher}}} \log D_j(x) + \mathbb{E}_{x \sim p_{\text{student}}} \log (1 - D_j(x)) \quad (8)$$

where $x \sim p_{\text{student}}$ are outputs from generation network $S_\theta$. At the same time, $S_\theta$ attempts to generate similar outputs which will fool the discriminator by minimizing $\mathbb{E}_{x \sim p_{\text{student}}} \log (1 - D_j(x))$.

In Eq. 9, $x$ is the concatenation of teacher and student outputs. We feed $x$ into the discriminator which is a three-layer fully-connected network. The whole structure of a discriminator is shown in Fig. 6. **Multi-Stage Discriminators.** Using multi-Stage discriminators can refine the student outputs gradually. As shown in Fig. 4, the final adversarial loss is a sum of the individual ones (by minimizing $\mathcal{L}^j_{GAN}$):

$$\mathcal{L}_{GAN} = -\sum_{j \in A} \mathcal{L}^j_{GAN} \quad (9)$$

Let $|A|$ be the number of discriminators. In our experiments, we use 3 for CIFAR [59] and SVHN [60], and 5 for ImageNet [1].

### 5 Learning Strategy of Similarity and Discriminators

#### 5.1 Joint Training

For the strategy of joint training, we incorporate the similarity loss in Eq. 7 and adversarial loss in Eq. 9 into our final loss function based on above definition and analysis. Our whole framework is trained end-to-end by the following objective function:

$$\mathcal{L} = \alpha \mathcal{L}_{Sim} + \beta \mathcal{L}_{GAN} \quad (10)$$

where $\alpha$ and $\beta$ are trade-off weights. We set them as 1 in our experiments by cross validation. We also use the weighted coefficients to balance the contributions of different blocks. For 3-block networks, we use $[0.01, 0.05, 1]$, and $[0.001, 0.01, 0.05, 0.1, 1]$ for 5-block ones.

#### 5.2 Alternately Update Gradients

For alternately updating gradients, we follow the training process of standard generative adversarial networks [22] which trains $D$ to maximize the probability of assigning
Fig. 7: Illustration of our two gradient update strategies. Red line indicates *joint training strategy* and black dash line indicates *alternate updating strategy*. More details can be referred to Section 5.

**Algorithm 1 Learning Strategy of Similarity and Discriminators.**

**Strategy 1: Alternately Update Gradients.**

Require: Following [22], we also define the number of steps to apply to the discriminator, \(k\), as a hyperparameter. \(\alpha\) and \(\beta\) are the trade-off coefficients.

1: for number of training iterations do:
2: for \(k\) steps do:
3: Sample minibatch of \(m\) examples \(X_1, \ldots, X_m\) from training data through teacher as distribution \(p_T(T_\theta(X_i))\) and through student as distribution \(p_S(S_\theta(X_i))\).
4: Update the j-th discriminator \(D\) by ascending its stochastic gradient:
   \[
   \nabla_{\theta_D} \frac{1}{m} \sum_{i=1}^{m} [\log D(T_\theta(X_i)) + \log (1 - D(S_\theta(X_i)))]
   \]
5: end for
6: Sample minibatch of \(m\) examples \(X_1, \ldots, X_m\) from training data through student as distribution \(p_S(S_\theta(X_i))\).
7: Update student by descending its stochastic gradient:
   \[
   \nabla_{\theta_S} \frac{1}{m} \sum_{i=1}^{m} [\alpha \mathcal{L}_{Sim} + \beta \log (1 - D(S_\theta(X_i)))]
   \]
8: end for

**Strategy 2: Joint Training.**

1: for number of training iterations do:
2: Sample minibatch of \(m\) examples \(X_1, \ldots, X_m\) from training data distribution \(p_{data}(X)\).
3: Update the j-th discriminator \(D\) and the student by descending its stochastic gradient:
   \[
   \nabla_{\theta_D, \theta_S} \frac{1}{m} \sum_{i=1}^{m} [\alpha \mathcal{L}_{Sim} + \beta \mathcal{L}_{GAN}]
   \]
4: end for

The gradient-based updates can use any standard gradient-based learning rule.

the correct label to both teacher features and features from student. We simultaneously train \(G\) (student) to minimize \(\log(1 - D(G(x)))\). As in [22], \(D\) and \(G\) play the following two-player minimax game with value function \(V(G, D)\):

\[
\min_G \max_D V(G, D) = \mathbb{E}_{T_\theta(X_i) \sim P_T} [\log D(T_\theta(X_i))] + \mathbb{E}_{S_\theta(X_i) \sim P_S} [\log (1 - D(S_\theta(X_i)))]
\]

(11)

We update the student network after updating the discriminator in \(k\) iterations (we choose \(k = 1\) in all our experiments). When updating the student network \(S_\theta\), we aim to fool the discriminator by fixing discriminator \(D\) and minimizing the similarity loss \(\mathcal{L}_{Sim}\) and GAN loss. More details can be referred in Algorithm 1 and Fig. 7.

The main difference between *joint training* and *alternate updating* is that in the former strategy, the gradients from discriminators will propagate back to the student backbone, while in the latter strategy, the parameters in discriminators and student will update separately without any interactions.

6 MULTI-MODEL ENSEMBLE VIA ADVERSARIAL LEARNING (MEAL)

We achieve ensemble with a training method that is simple and straight-forward to implement. As different network structures can obtain different distributions of outputs, which can be viewed as soft labels (knowledge), we adopt these soft labels to train our student, in order to compress knowledge of different architectures into a single network. Thus we can obtain the seemingly contradictory goal of ensembling multiple neural networks at no additional testing cost.

6.1 Learning Procedure

To clearly understand what the student learned in our work, we define two conditions. First, the student has the same structure as the teacher network. Second, we choose one structure for student and randomly select a structure for teacher in each iteration as our ensemble learning procedure.

The learning procedure contains two stages. First, we pre-train the teachers to produce a model zoo. Because we use the classification task to train these models, we can use the softmax cross entropy loss as the main training loss.
Fig. 8: Image samples from two benchmarks grouped by clean and noisy/multi-label categories. In each group, black box images are clean labeled images and red box images are ones with multi-label objects (ImageNet) or noisy label images (iMaterialist products).

Algorithm 2 Multi-Model Ensemble via Adversarial Learning (MEAL).

Stage 1: Building and Pre-training the Teacher Model Zoo $\mathcal{T} = \{T_1^\theta, T_2^\theta, \ldots, T_i^\theta\}$, including: VGGNet [61], ResNet [62], DenseNet [63], MobileNet [64], Shake-Shake [8], etc.

Stage 2:
1: function $TSM(T)$
2: $T_\theta \leftarrow R(S(T))$ \hspace{1cm} $\triangleright$ Random Selection
3: return $T_\theta$
4: end function
5: for each iteration do:
6: $T_\theta \leftarrow TSM(T)$ \hspace{1cm} $\triangleright$ Randomly Select a Teacher Model
7: $S_\theta = \arg\min_{S_\theta} L(T_\theta, S_\theta)$ \hspace{1cm} $\triangleright$ Adversarial Learning for a Student
8: end for

7 LEARNING ON NOISY DATA

Deep neural networks have achieved notable success in image classification recently due to the collection of massive large-scale labeled datasets such as ImageNet [1], OpenImage [2], etc. However, collecting such datasets is time-consuming and expensive, further requires double check from multiple annotators to reduce label error. So a better solution is to automatically build and learn from an Internet-scale dataset with noisy labels. Our method is fairly easy to extend to handle noisy data with an automatic way, since the “soft labels” predicted from teacher models usually are more accurate than the noisy labels provided by the noisy dataset. We further propose an iterative refinement strategy to boost the performance of our method on noisy labeled dataset.

7.1 Iterative Refinement

We propose to use an iterative training strategy to refine the noisy labels, which mainly exhibits three advantages: (1) Rectify sample supervisions with potentially wrong class labels through the teacher model predictions; (2) Improve the quality of predictions from the teacher model $\mathcal{T}$ through the Iterative Refinement, so that the similarity loss will be more effective; And (3) Mitigate the overfit to noisy samples when training networks on noisy labeled data, which leads to more robustness of student models against label noise.

Firstly, we perform an initial training iteration following the method described in Algorithm 2, and obtain a model with the best validation accuracy. This model will be the teacher in the next training iteration. In the second training iteration, we repeat the steps in Algorithm 2 with only one change described as follows. We replace the teacher model zoo with the models we learned from the first step. This operation improve the quality of teacher models and promote the teacher model to produce more reliable predictions for student model training, which can improve the quality of student models.

8 EXPERIMENTS AND ANALYSIS

We empirically demonstrate the effectiveness of MEAL on several benchmark datasets. We implement our method on the PyTorch [65] platform.

8.1 Datasets

CIFAR. The two CIFAR datasets [59] consist of colored natural images with a size of $32 \times 32$. CIFAR-10 is drawn from 10 and CIFAR-100 is drawn from 100 classes. In each dataset, the train and test sets contain 50,000 and 10,000 images, respectively. A standard data augmentation scheme $^3$ [66],

3. zero-padded with 4 pixels on both sides, randomly cropped to produce $32 \times 32$ images, and horizontally mirror with probability 0.5.
[67], [68], [10], [69] is used. We report the test errors in this section with training on the whole training set.

**SVHN.** The Street View House Number (SVHN) dataset [60] consists of 32 × 32 colored digit images, with one class for each digit. The train and test sets contain 604,388 and 26,032 images, respectively. Following previous works [70], [5], [10], [69], we split a subset of 6,000 images for validation, and train on the remaining images without data augmentation.

**ImageNet.** The ILSVRC 2012 classification dataset [1] consists of 1000 classes, with a number of 1.2 million training images and 50,000 validation images. We adopt the the data augmentation scheme following [71] and apply the same operation as [10] at test time.

**iMaterialist Challenge Dataset**. The iMaterialist Dataset contains about one million product images with 2019 classes for training, about 10K images for validation and 90K images for testing. This dataset is fairly challenging since about 30% training images are with incorrect labels. We follow the evaluation in the competition which uses top-3 classification error as metric. We also provide top-1 results in our experiments.

### 8.2 Networks

We adopt several popular network architectures as our teacher model zoo, including VGGNet [61], ResNet [62], DenseNet [63], MobileNet [64], shake-shake [8], etc. For VGGNet, we use 19-layer with Batch Normalization [72]. For ResNet, we use 18-layer network for CIFAR and SVHN and 50-layer for ImageNet. For DenseNet, we use the BC structure with depth L=100, and growth rate k=24. For shake-shake, we use 26-layer 2×96d version. Note that due to the high computing costs, we use shake-shake as a teacher only when the student is shake-shake network.

### 8.3 Ablation Studies

We first investigate each design principle of our MEAL framework with joint training strategy. We design several controlled experiments on CIFAR-10 with VGGNet-19 w/BN (both to teacher and student) for this ablation study. A consistent setting is imposed on all the experiments, unless when some components or structures are examined.

The results are mainly summarized in Table 1. The first three rows indicate that we only use \( \ell_1, \ell_2 \) or cross-entropy loss from the last layer of a network. It’s similar to the Knowledge Distillation method. We can observe that use cross-entropy achieve the best accuracy. Then we employ more intermediate outputs to calculate the loss, as shown in rows 4 and 5. It’s obvious that including more layers improves the performance. Finally, we involve the discriminators to exam the effectiveness of adversarial learning. Using cross-entropy, intermediate layers and adversarial learning achieve the best result. Additionally, we use average based adaptive pooling for alignment. We also tried max operation, the accuracy is much worse (6.32%).

**8.4 Results of Multi-Model Ensemble**

**Comparison with Different Learning Strategy.** We compare MEAL with joint update and alternate update strategies. The results are shown in Table 2, we employ several network architectures in this comparison. All models are trained with the same epochs. It can be observed that in the most cases joint training obtains better performance than alternate update on all the networks, so in the following experiments, we use joint update as our basic learning method unless otherwise noted.

**TABLE 2:** Comparison of error rate (%) with different learning strategies on CIFAR-10.

| Network          | Alternate Updating (A) (%) | Joint Training (J) (%) |
|------------------|-----------------------------|------------------------|
| VGG-19 w/BN [61] | 5.87                        | 5.55                   |
| GoogLeNet [73]   | 4.39                        | 4.83                   |
| ResNet-18 [62]   | 4.43                        | 4.35                   |
| DenseNet-BC (k=24) [63] | 3.99 | 3.54 |

**Comparison with Traditional Ensemble.** The results are summarized in Fig. 9 and Table 3. In Figure 9, we compare
the error rate using the same architecture on a variety of datasets (except ImageNet). It can be observed that our results consistently outperform the single and traditional methods on these datasets. The traditional ensembles are obtained through averaging the final predictions across all teacher models. In Table 3, we compare error rate using different architectures on the same dataset. In most cases, our ensemble method achieves lower error than any of the baselines, including the single model and traditional ensemble.

**TABLE 3:** Error rate (%) using different network architectures on CIFAR-10 dataset.

| Network               | Single (%) | Traditional Ens. (%) | Our Ens. (%) |
|-----------------------|------------|----------------------|--------------|
| MobileNet [64]        | 10.70      | -                    | 8.09         |
| VGG-19 w/ BN [61]     | 6.34       | -                    | 5.55         |
| DenseNet-BC (k=24) [63]| 3.76       | 3.73                 | 3.54         |
| Shake-Shake-26 2x96d [6] | 2.86       | 2.79                 | 2.54         |

**Comparison with Dropout.** We compare MEAL with the “Implicit” method Dropout [3]. The results are shown in Table 4, we employ several network architectures in this comparison. All models are trained with the same epochs. We use a probability of 0.2 for drop nodes during training. It can be observed that our method achieves better performance than Dropout on all these networks.

**TABLE 4:** Comparison of error rate (%) with Dropout [3] baseline on CIFAR-10.

| Network               | Dropout (%) | Our Ens. (%) |
|-----------------------|-------------|--------------|
| VGG-19 w/ BN [61]     | 6.89        | 5.55         |
| GoogLeNet [73]        | 5.37        | 4.83         |
| ResNet-18 [62]        | 4.69        | 4.35         |
| DenseNet-BC (k=24) [63]| 3.75       | 3.54         |

**TABLE 5:** Val. error (%) on ImageNet dataset.

| Method                | Top-1 (%) | Top-5 (%) | #FLOPs | Inference Time (per/image) |
|-----------------------|-----------|-----------|--------|----------------------------|
| VGG-19 w/BN           | 25.76     | 8.15      | 19.32B | $5.70 \times 10^{-3}$      |
| ResNet-50             | 23.85     | 7.13      | 4.09B  | $1.10 \times 10^{-3}$      |
| Ours (ResNet-50)      | 23.58     | 6.86      | 4.09B  | $1.10 \times 10^{-3}$      |
| Traditional Ens.      | 22.76     | 6.49      | 23.65B | $1.67 \times 10^{-3}$      |
| Ours Plus J (ResNet-50)| 21.79     | 5.99      | 4.09B  | $1.10 \times 10^{-3}$      |
| Ours Plus A (ResNet-50)| 22.08     | 5.93      | 4.09B  | $1.10 \times 10^{-3}$      |

**Our Learning-Based Ensemble Results on ImageNet.** As shown in Table 5, we compare our ensemble method with the original model and the traditional ensemble. We use VGG-19 w/BN and ResNet-50 as our teachers, and use ResNet-50 as the student. The #FLOPs and inference time for traditional ensemble are the sum of individual ones. Therefore, our method has both better performance and higher efficiency. Most notably, our MEAL plus yields an error rate of Top-1 21.79%, Top-5 5.99% on ImageNet, far outperforming the original ResNet-50 23.85%/7.13% and the traditional ensemble 22.76%/6.49%. This shows great potential on large-scale real-size datasets.

**8.5 Results of Noisy Data Refinement**

**Base Model Training (Teachers).** We first train our base models following the parameter-setting from ImageNet with three network structures: VGG-19, ResNet-50 and ResNet-101. In particular, we use the ImageNet pre-trained networks as initial parameters and the initial learning rate is set to 0.01, and then divided by 10 after 45 epochs. The total training budget is 60 epochs. The whole training error curves are illustrated in Fig. 10. We can observe that because of the large percentage of noisy labels in the training set, the training errors are higher than that on validation set (validation and testing sets are cleaned manually by the organizers). The results on testing set are show in Table 6.
The baseline result is 8.96%, our MEAL outperforms the baseline by 1.15% (8.96% vs. 7.81%). We further adopt recently proposed data augmentation method [74], [75] to verify whether our model overfits to the training data. From Table 6 we can see that after using CutMix [75] and Cosine Learning Rate schedule [76], our result further improves to 6.89%, which demonstrates that our model doesn’t overfit to the noisy training data and still has space to improve.

**Iterative Refinement.** Then we iteratively refine our MEAL model with the strategy we described above. Every time we replace the teacher model from the previous round, which can generate better probabilities as supervision for the student model training. We show the train accuracy curves of first and second re-training rounds in Fig. 11. It is obvious that second re-training has better performance than the first re-training, which verifies the effectiveness of our iterative refinement strategy.

### 8.6 Analysis

**Effectiveness of Ensemble Size.** Fig. 12 displays the performance of three architectures on CIFAR-10 as the ensemble size is varied. Although ensembling more models generally gives better accuracy, we have two important observations. First, we observe that our single model “ensemble” already outputs the baseline model with a remarkable margin, which demonstrates the effectiveness of adversarial learning. Second, we observe some drops in accuracy using the VGGNet and DenseNet networks when including too many ensembles for training. In most case, an ensemble of four models obtains the best performance.

**Budget for Training.** On CIFAR datasets, the standard training budget is 300 epochs. Intuitively, our ensemble method can benefit from more training budget, since we use the diverse soft distributions as labels. Fig. 14 displays the relation between performance and training budget. It appears that more than 400 epochs is the optimal choice and our model will fully converge at about 500 epochs.

**Effectiveness of Re-training Number.** Fig. 13 displays the performance of three networks on iMaterialist products dataset as the number of re-training is varied. We can observe that the first re-training process generally gives most
We have presented MEAL, a learning-based ensemble method that can compress multi-model knowledge into a single network with adversarial learning. Our experimental evaluation on three benchmarks CIFAR-10/100, SVHN, ImageNet and iMaterialist Products Dataset verified the effectiveness of our proposed method, which achieved the state-of-the-art accuracy for a variety of network architectures. Our further work will focus on adopting MEAL for cross-domain ensemble and adaption.

References

[1] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, “Imagenet: A large-scale hierarchical image database,” in 2009 IEEE conference on computer vision and pattern recognition. Ieee, 2009, pp. 248–255.

[2] I. Krasin, T. Duerig, N. Alldrin, A. Veit et al., “Openimages: A public dataset for large-scale multi-label and multi-class image classification.” https://github.com/openimages, 2016.

[3] N. Srivastava, G. E. Hinton, A. Krizhevsky et al., “Dropout: a simple way to prevent neural networks from overfitting.” JMLR, 2014.

[4] L. Wan, M. Zeiler, S. Zhang, Y. Le Cun, and R. Fergus, “Regularization of neural networks using dropconnect,” in International conference on machine learning, 2013, pp. 1058–1066.

[5] G. Huang, Y. Sun, Z. Liu, D. Sedra, and K. Q. Weinberger, “Deep networks with stochastic depth,” in European conference on computer vision. Springer, 2016, pp. 646–661.

[6] S. Singh, D. Hoiem, and D. Forsyth, “Swapout: Learning an ensemble of deep architectures,” in Advances in neural information processing systems, 2016, pp. 28–36.

[7] L. v. d. Maaten and G. Hinton, “Visualizing data using t-sne,” Journal of machine learning research, vol. 9, no. Nov, pp. 2579–2605, 2008.

[8] X. Gastaldi, “Shake-shake regularization,” arXiv preprint arXiv:1705.07485, 2017.

[9] Z. Shen, Z. He, and X. Xue, “Meal: Multi-model ensemble via adversarial learning,” in AAAI, 2019.

[10] G. Huang, Y. Li, G. Pleiss, Z. Liu, J. E. Hopcroft, and K. Q. Weinberger, “Snapshot ensembles: Train 1, get m for free,” in ICLR, 2017.

[11] L. K. Hansen and P. Salamon, “Neural network ensembles,” IEEE transactions on pattern analysis and machine intelligence, vol. 12, no. 10, pp. 993–1001, 1990.

[12] M. P. Perrone and L. N. Cooper, “When networks disagree: Ensemble methods for hybrid neural networks,” in How We Learn; How We Remember: Toward an Understanding of Brain and Neural Systems: Selected Papers of Leon N Cooper. World Scientific, 1995, pp. 342–358.

[13] A. Krogh and J. Vedelsby, “Neural network ensembles, cross validation, and active learning,” in Advances in neural information processing systems, 1995, pp. 231–238.

[14] T. G. Dietterich, “Ensemble methods in machine learning,” in International workshop on multiple classifier systems, 2000, pp. 1–15.

[15] B. Lakshminarayanan, A. Pritzel, and C. Blundell, “Simple and scalable predictive uncertainty estimation using deep ensembles,” in Advances in Neural Information Processing Systems, 2017, pp. 6402–6413.

[16] X. Zhu, S. Gong et al., “Knowledge distillation by on-the-fly naive ensemble,” in Advances in Neural Information Processing Systems, 2018, pp. 7517–7527.

[17] T. Zhou, S. Wang, and J. A. Bilmes, “Diverse ensemble evolution: Curriculum data-model marriage,” in Advances in Neural Information Processing Systems, 2018, pp. 5905–5916.

[18] G. Hinton, O. Vinyals, and J. Dean, “Distilling the knowledge in a neural network,” arXiv preprint arXiv:1503.02531, 2015.

[19] N. Papernot, M. Abadi, U. Erlingsson, I. Goodfellow, and K. Talwar, “Semi-supervised knowledge transfer for deep learning from private training data,” in ICLR, 2017.

[20] Y. Li, J. Yang, Y. Song, L. Cao, J. Luo, and L.-J. Li, “Learning from noisy labels with distillation,” in Proceedings of the IEEE International Conference on Computer Vision, 2017, pp. 1910–1918.

[21] J. Yim, D. Joo, J. Bae, and J. Kim, “A gift from knowledge distillation: Fast optimization, network minimization and transfer learning,” in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 4133–4141.

[22] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, “Generative adversarial nets,” in Advances in neural information processing systems, 2014, pp. 2672–2680.

[23] H. Bagherinezhad, M. Horton, M. Rastegari, and A. Farhadi, “Label refiner: Improving imagenet classification through label progression,” in ECCV, 2018.
[73] C. Szegedy, W. Liu, Y. Jia, P. Sermanet et al., “Going deeper with convolutions,” in CVPR, 2015.

[74] H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-Paz, “mixup: Beyond empirical risk minimization,” in International Conference on Learning Representations, 2018.

[75] S. Yun, D. Han, S. J. Oh, S. Chun, J. Choe, and Y. Yoo, “Cutmix: Regularization strategy to train strong classifiers with localizable features,” arXiv preprint arXiv:1905.04899, 2019.

[76] I. Loshchilov and F. Hutter, “Sgdr: Stochastic gradient descent with warm restarts,” arXiv preprint arXiv:1608.03983, 2016.