# MEAL: Multi-Model Ensemble via Adversarial Learning

**Zhiqiang Shen[1,2*], Zhankui He[3*], Xiangyang Xue[1]**
[1]Shanghai Key Laboratory of Intelligent Information Processing,
School of Computer Science, Fudan University, Shanghai, China
[2]Beckman Institute, University of Illinois at Urbana-Champaign, IL, USA
[3]School of Data Science, Fudan University, Shanghai, China
zhiqiangshen0214@gmail.com, {zkhe15, xyxue}@fudan.edu.cn

### Abstract

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 optimized better 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 student network can learn the distilled knowledge from a teacher model that has arbitrary structures. Extensive experiments on CIFAR-10/100, SVHN and ImageNet datasets demonstrate the effectiveness of our MEAL method. On ImageNet, our ResNet-50 based MEAL achieves top-1/5 21. 79

## 1. Introduction

The 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. This benefit has also been exploited indirectly when training a single network through Dropout (Srivastava et al. 2014), Dropconnect (Wan et al. 2013), Stochastic Depth (Huang et al. 2016), Swapout (Singh, Hoiem, and Forsyth 2016), 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 pre-trained networks correspond to an ensemble prediction of a group of individual reference networks.

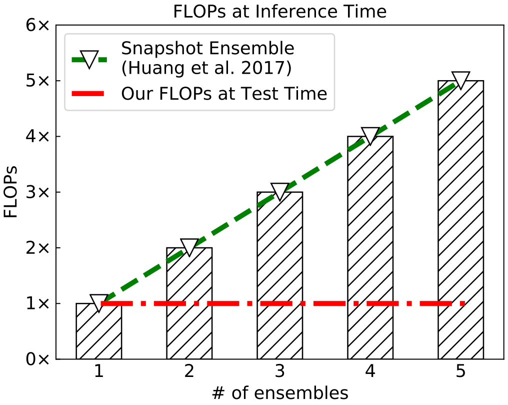

Figure 1: Comparison of FLOPs at inference time. Huang et al. (Huang et al. 2017a) employ models at different local minimum for ensembling, which enables no additional training cost, but the computational FLOPs at test time linearly increase with more ensembles. In contrast, our method use only one model during inference time throughout, so the testing cost is independent of # ensembles.

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*. 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. 2, an image of "tobacco shop" has similar appearance to "library" should have a different la-

[*]Equal contribution. This work was done when Zhankui He was a research intern at University of Illinois at Urbana-Champaign.

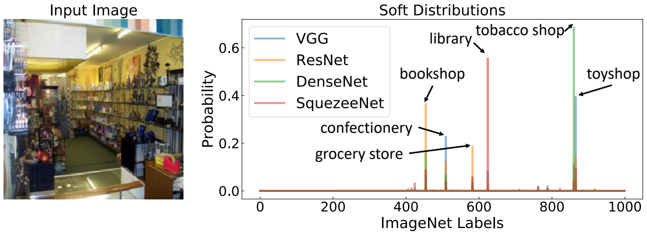

Figure 2: Left is a training example of class "tobacco shop" from ImageNet. Right are soft distributions from different trained architectures. The soft labels are more *informative* and can provide more coverage for visually-related scenes.

bel 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. Our experiments show that MEAL consistently improves the accuracy across a variety of popular network architectures on different datasets. For instance, our shake-shake (Gastaldi 2017) based MEAL achieves 2.54% test error on CIFAR-10, which is a relative 11.2% improvement[1]. On ImageNet, our ResNet-50 based MEAL achieves 21.79%/5.99% val error, which outperforms the baseline by a large margin.

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.
- 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 for a variety of existing network architectures.

## 2. Related Work

There is a large body of previous work (Hansen and Salamon 1990; Perrone and Cooper 1995; Krogh and Vedelsby 1995; Dietterich 2000; Huang et al. 2017a; Lakshminarayanan, Pritzel, and Blundell 2017) on ensembles with neural networks. However, most of these prior studies focus on improving the generalization of an individual network. Recently, Snapshot Ensembles (Huang et al. 2017a) 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 (Hinton, Vinyals, and Dean 2015; Papernot et al. 2017; Li et al. 2017; Yim et al. 2017) and adversarial learning (Goodfellow et al. 2014), 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 (Srivastava et al. 2014), DropConnection (Wan et al. 2013), Stochastic Depth (Huang et al. 2016), Swapout (Singh, Hoiem, and Forsyth 2016), 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 (Bagherinezhad et al. 2018), 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.

**Adversarial Learning.** Generative Adversarial Learning (Goodfellow et al. 2014) is 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. 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 generation (Johnson, Gupta, and Fei-Fei 2018), detection (Bai et al. 2018), etc.

**Knowledge Transfer.** Distilling knowledge from trained neural networks and transferring it to another new network has been well explored in (Hinton, Vinyals, and Dean 2015; Chen, Goodfellow, and Shlens 2016; Li et al. 2017; Yim et al. 2017; Bagherinezhad et al. 2018; Anil et al. 2018). 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. (Yim et al. 2017) define the distilled knowledge to be transferred flows between different intermediate layers and computer the inner product between parameters from two networks. Bagherinezhad et al. (Bagherinezhad et al. 2018) 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.

## 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. 3. 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

[1] Shake-shake baseline (Gastaldi 2017) is 2.86%.

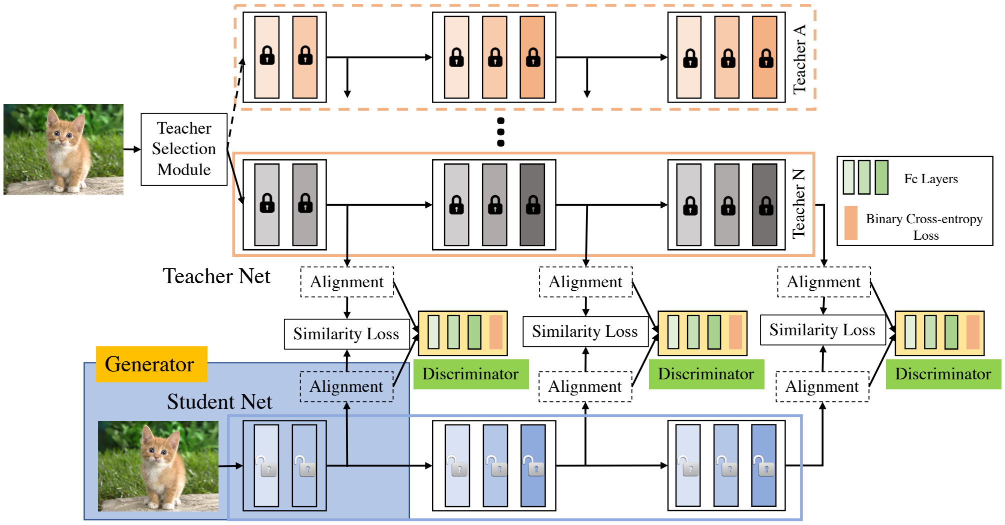

Figure 3: Overview of our proposed architecture. We input the same image into the teacher and student networks to generate intermediate and final outputs for *Similarity Loss* and *Discriminators*. The model is trained adversarially against several *discriminator networks*. During training the model observes supervisions from trained teacher networks instead of the one-hot ground-truth labels, and the teacher's parameters are fixed all the time.

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

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

$$\begin{aligned}\mathcal{L}_{Sim} &= d(\mathcal{T}_\theta(X_i), \mathcal{S}_\theta(X_i)) \\ &= \sum_c d(p_c^{\mathcal{T}_\theta}(X_i), p_c^{\mathcal{S}_\theta}(X_i))\end{aligned} \quad (1)$$

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:

$$\mathcal{L}_{\ell_1\_Sim}(\mathcal{S}_\theta) = \frac{1}{n}\sum_c\sum_{i=1}^{n}\left|p_c^{\mathcal{T}_\theta}(X_i) - p_c^{\mathcal{S}_\theta}(X_i)\right|^1 \quad (2)$$

$\ell_2$ **distance** or euclidean distance is the straight-line distance in euclidean space. 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:

$$\mathcal{L}_{\ell_2\_Sim}(\mathcal{S}_\theta) = \frac{1}{n}\sum_c\sum_{i=1}^{n}\left\|p_c^{\mathcal{T}_\theta}(X_i) - p_c^{\mathcal{S}_\theta}(X_i)\right\|^2 \quad (3)$$

**KL-divergence** is a measure of how one probability distribution is different from another reference probability distribution. Here we train student network $\mathcal{S}_\theta$ by minimizing the KL-divergence between its output $p_c^{\mathcal{S}_\theta}(X_i)$ and the soft labels $p_c^{\mathcal{T}_\theta}(X_i)$ generated by the teacher network. Our loss function is:

$$\begin{aligned}\mathcal{L}_{KL\_Sim}(\mathcal{S}_\theta) &= -\frac{1}{n}\sum_c\sum_{i=1}^{n} p_c^{\mathcal{T}_\theta}(X_i)\log(\frac{p_c^{\mathcal{S}_\theta}(X_i)}{p_c^{\mathcal{T}_\theta}(X_i)}) \\ &= -\frac{1}{n}\sum_c\sum_{i=1}^{n} p_c^{\mathcal{T}_\theta}(X_i)\log p_c^{\mathcal{S}_\theta}(X_i) \\ &\quad +\frac{1}{n}\sum_c\sum_{i=1}^{n} p_c^{\mathcal{T}_\theta}(X_i)\log p_c^{\mathcal{T}_\theta}(X_i)\end{aligned} \quad (4)$$

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

$$\mathcal{L}_{CE\_Sim}(\mathcal{S}_\theta) = -\frac{1}{n}\sum_c\sum_{i=1}^{n} p_c^{\mathcal{T}_\theta}(X_i)\log p_c^{\mathcal{S}_\theta}(X_i) \quad (5)$$

[2] Ground-truth labels

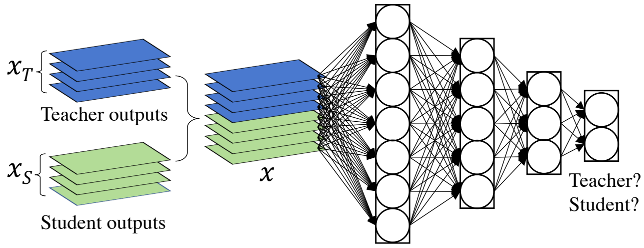

Figure 4: Illustration of our proposed discriminator. We concatenate the outputs of teacher and student as the inputs of a discriminator. The discriminator is a three-layer fully-connected network.

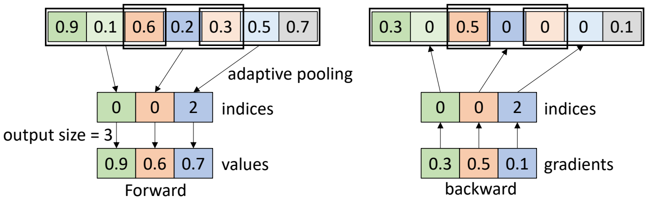

Figure 5: The process of adaptive pooling in forward and backward stages. We use max operation for illustration.

### 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:

$$\mathcal{L}^j_{Sim} = d(f(\mathcal{T}_{\theta_j}), f(\mathcal{S}_{\theta_j})) \tag{6}$$

where $T_{\theta_j}$ and $S_{\theta_j}$ 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. 5 illustrates the process of adaptive pooling. Because we adopt multiple intermediate layers, our final similarity loss is a sum of individual one:

$$\mathcal{L}_{Sim} = \sum_{j \in \mathcal{A}} \mathcal{L}^j_{Sim} \tag{7}$$

where $\mathcal{A}$ 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 $\mathcal{S}_\theta$ 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 (Goodfellow et al. 2014):

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

where $x \sim p_{\text{student}}$ are outputs from generation network $\mathcal{S}_{\theta_j}$. At the same time, $\mathcal{S}_{\theta_j}$ 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. 4.

**Multi-Stage Discriminators.** Using multi-Stage discriminators can refine the student outputs gradually. As shown in Fig. 3, the final adversarial loss is a sum of the individual ones (by minimizing -$\mathcal{L}^j_{GAN}$):

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

Let $|\mathcal{A}|$ be the number of discriminators. In our experiments, we use 3 for CIFAR (Krizhevsky 2009) and SVHN (Netzer et al. 2011), and 5 for ImageNet (Deng et al. 2009).

### 4.4 Joint Training of Similarity and Discriminators

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

$$\mathcal{L} = \alpha \mathcal{L}_{Sim} + \beta \mathcal{L}_{GAN} \tag{10}$$

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 ues [0.01, 0.05, 1], and [0.001, 0.01, 0.05, 0.1, 1] for 5-block ones.

## 5. 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*.

### 5.1 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 pretrain 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 in this stage. Second, we minimize the loss function $\mathcal{L}$ in Eq. 10 to make the student output similar to that of the teacher output. The learning procedure is explained below in Algorithm 1.

## 6. Experiments and Analysis

We empirically demonstrate the effectiveness of MEAL on several benchmark datasets. We implement our method on the PyTorch (Paszke et al. 2017) platform.

**Algorithm 1** Multi-Model Ensemble via Adversarial Learning (MEAL).

**Stage 1:**

Building and Pre-training the Teacher Model Zoo $\mathcal{T} = \{\mathcal{T}_\theta^1, \mathcal{T}_\theta^2, \dots \mathcal{T}_\theta^i\}$, including: VGGNet (Simonyan and Zisserman 2015), ResNet (He et al. 2016), DenseNet (Huang et al. 2017b), MobileNet (Howard et al. 2017), Shake-Shake (Gastaldi 2017), etc.

**Stage 2:**

```
function $TSM(\mathcal{T})$
    $\mathcal{T}_\theta \leftarrow RS(\mathcal{T})$                ▷ Random Selection
    return $\mathcal{T}_\theta$
end function
for each iteration do:
    $\mathcal{T}_\theta \leftarrow TSM(\mathcal{T})$        ▷ Randomly Select a Teacher Model
    $\mathcal{S}_\theta = \arg\min_{\mathcal{S}_\theta} \mathcal{L}(\mathcal{T}_\theta, \mathcal{S}_\theta)$   ▷ Adversarial Learning for a Student
end for
```

## 6.1. Datasets

**CIFAR.** The two CIFAR datasets (Krizhevsky 2009) consist of colored natural images with a size of 32×32. 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] (Lee et al. 2015; Romero et al. 2015; Larsson, Maire, and Shakhnarovich 2016; Huang et al. 2017a; Liu et al. 2017) 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 (Netzer et al. 2011) 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 (Goodfellow et al. 2013; Huang et al. 2016; 2017a; Liu et al. 2017), 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 (Deng et al. 2009) 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 (Krizhevsky, Sutskever, and Hinton 2012) and apply the same operation as (Huang et al. 2017a) at test time.

## 6.2 Networks

We adopt several popular network architectures as our teacher model zoo, including VGGNet (Simonyan and Zisserman 2015), ResNet (He et al. 2016), DenseNet (Huang et al. 2017b), MobileNet (Howard et al. 2017), shake-shake (Gastaldi 2017), etc. For VGGNet, we use 19-layer with Batch Normalization (Ioffe and Szegedy 2015). For ResNet, we use 18-layer network for CIFAR and SVHN and 50-layer for ImagNet. 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.

Table 1: Ablation study on CIFAR-10 using VGGNet-19 w/BN. Please refer to Section 6.3 for more details.

| $\ell_1$ dis. | $\ell_2$ dis. | Cross-Entropy | Intermediate | Adversarial | Test Errors (%) |
|---|---|---|---|---|---|
| **Base Model (VGG-19 w/ BN) (Simonyan and Zisserman 2015)** | | | | | **6.34** |
| ✓ | | | | | 6.97 |
| | ✓ | | | | 6.22 |
| | | ✓ | | | 6.18 |
| | | ✓ | ✓ | | 6.10 |
| | ✓ | | ✓ | | 6.17 |
| | | ✓ | ✓ | ✓ | **5.83** |
| ✓ | | | ✓ | ✓ | 7.57 |

## 6.3 Ablation Studies

We first investigate each design principle of our MEAL framework. 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%).

## 6.4 Results

**Comparison with Traditional Ensemble.** The results are summarized in Figure 6 and Table 2. In Figure 6, 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 2, 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 2: Error rate (%) using different network architectures on CIFAR-10 dataset.

| Network | Single (%) | Traditional Ens. (%) | Our Ens. (%) |
|---|---|---|---|
| MobileNet (Howard et al. 2017) | 10.70 | – | **8.09** |
| VGG-19 w/ BN (Simonyan and Zisserman 2015) | 6.34 | – | **5.55** |
| DenseNet-BC ($k$=24) (Huang et al. 2017b) | 3.76 | 3.73 | **3.54** |
| Shake-Shake-26 2x96d (Gastaldi 2017) | 2.86 | 2.79 | **2.54** |

**Comparison with Dropout.** We compare MEAL with the "Implicit" method Dropout (Srivastava et al. 2014). The results are shown in Table 3, 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.

[3] zero-padded with 4 pixels on both sides, randomly cropped to produce 32x32 images, and horizontally mirror with probability 0.5.

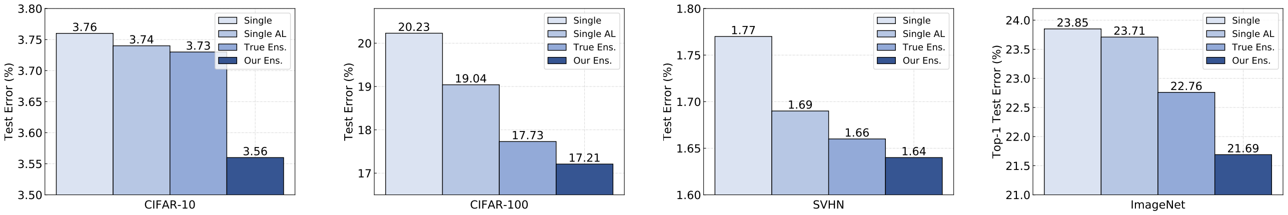

Figure 6: Error rates (%) on CIFAR-10 and CIFAR-100, SVHN and ImageNet datasets. In each figure, the results from left to right are 1) base model; 2) base model with adversarial learning; 3) true ensemble/traditional ensemble; and 4) our ensemble results. For the first three datasets, we employ DenseNet as student, and ResNet for the last one (ImageNet).

Table 3: Comparison of error rate (%) with Dropout (Srivastava et al. 2014) baseline on CIFAR-10.

| Network | Dropout (%) | Our Ens. (%) |
|---|---|---|
| VGG-19 w/ BN (Simonyan and Zisserman 2015) | 6.89 | **5.55** |
| GoogLeNet (Szegedy et al. 2015) | 5.37 | **4.83** |
| ResNet-18 (He et al. 2016) | 4.69 | **4.35** |
| DenseNet-BC ($k$=24) (Huang et al. 2017b) | 3.75 | **3.54** |

**Our Learning-Based Ensemble Results on ImageNet.** As shown in Table 4, 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[4] 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.

Table 4: **Val. error (%) on ImageNet dataset.**

| Method | Top-1 (%) | Top-5 (%) | #FLOPs | Inference Time (per/image) |
|---|---|---|---|---|
| **Teacher Networks:** | | | | |
| VGG-19 w/BN | 25.76 | 8.15 | 19.52B | $5.70 \times 10^{-3}$s |
| ResNet-50 | 23.85 | 7.13 | 4.09B | $1.10 \times 10^{-2}$s |
| Ours (ResNet-50) | 23.58 | 6.86 | 4.09B | $1.10 \times 10^{-2}$s |
| Traditional Ens. | 22.76 | 6.49 | 23.61B | $1.67 \times 10^{-2}$s |
| Ours Plus (ResNet-50) | **21.79** | **5.99** | **4.09B** | $\mathbf{1.10 \times 10^{-2}}$**s** |

## 6.5 Analysis

**Effectiveness of Ensemble Size.** Figure 8 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.

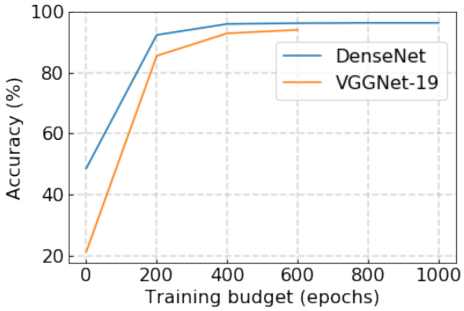

Figure 7: Accuracy of our ensemble method under different training budgets on CIFAR-10.

**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. Figure 7 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.

**Diversity of Supervision.** We hypothesize that different architectures create soft labels which are not only informative but also diverse with respect to object categories. We qualitatively measure this diversity by visualizing the pairwise correlation of softmax outputs from two different networks. To do so, we compute the softmax predictions for each training image in ImageNet dataset and visualize each pair of the corresponding ones. Figure 9 displays the bubble maps of four architectures. In the left figure, the coordinate of each bubble is a pair of $k$-th predictions ($p^k_{SequeezeNet}, p^k_{VGGNet}$), $k = 1, 2, \ldots, 1000$, and the right figure is ($p^k_{ResNet}, p^k_{DenseNet}$). If the label distributions are identical from two networks, the bubbles will be placed on the master diagonal. It's very interesting to observe that the left (weaker network pairs) has bigger diversity than the right (stronger network pairs). It makes sense because the stronger models generally tend to generate predictions close to the ground-truth. In brief, these differences in predictions can be exploited to create effective ensembles and our method is capable of improving the competitive baselines using this kind of diverse supervisions.

## 6.6 Visualization of the Learned Features

To further explore what our model actually learned, we visualize the embedded features from the single model and our ensembling model. The visualization is plotted by t-SNE tool (Maaten and Hinton 2008) with the last conv-layer features (2048 dimensions) from ResNet-50. We randomly sam-

[4] denotes using more powerful teachers like ResNet-101/152.

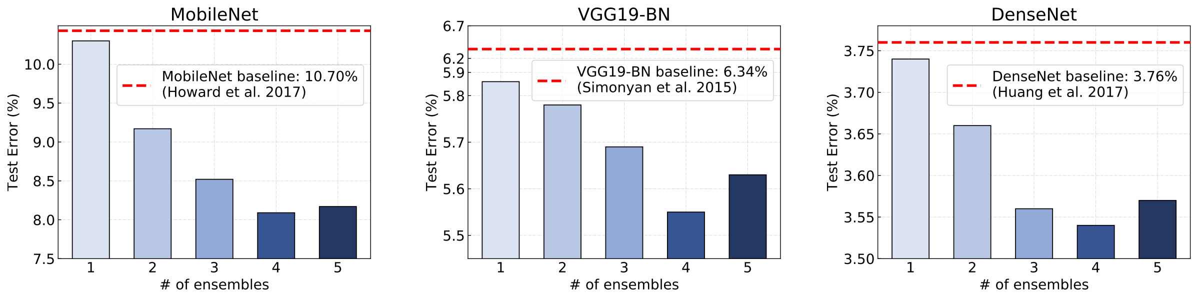

Figure 8: Error rate (%) on CIFAR-10 with MobileNet, VGG-19 w/BN and DenseNet.

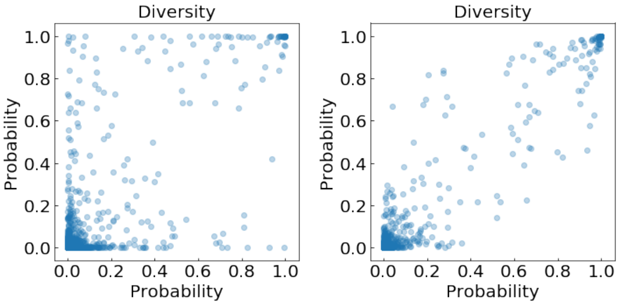

Figure 9: Probability Distributions between four networks. Left: SeqeezeNet *vs.* VGGNet. Right: ResNet *vs.* DenseNet.

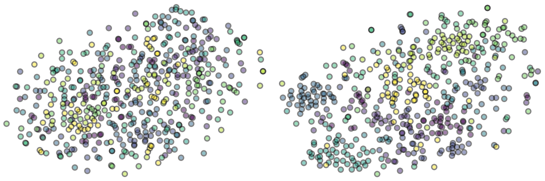

Figure 10: Visualizations of validation images from the ImageNet dataset by t-SNE (Maaten and Hinton 2008). We randomly sample 10 classes within 1000 classes. Left is the single model result using the standard training strategy. Right is our ensemble model result.

ple 10 classes on ImageNet, results are shown in Figure 10, it's obvious that our model has better feature embedding result.

## 7. Conclusion

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 and ImageNet 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.

**Acknowledgements** This work was supported in part by National Key R&D Program of China (No.2017YFC0803700), NSFC under Grant (No.61572138 & No.U1611461) and STCSM Project under Grant No.16JC1420400.

## References

Anil, R.; Pereyra, G.; Passos, A.; Ormandi, R.; Dahl, G. E.; and Hinton, G. E. 2018. Large scale distributed neural network training through online distillation. In *ICLR*.

Bagherinezhad, H.; Horton, M.; Rastegari, M.; and Farhadi, A. 2018. Label refinery: Improving imagenet classification through label progression. In *ECCV*.

Bai, Y.; Zhang, Y.; Ding, M.; and Ghanem, B. 2018. Finding tiny faces in the wild with generative adversarial network.

Chen, T.; Goodfellow, I.; and Shlens, J. 2016. Net2net: Accelerating learning via knowledge transfer. In *ICLR*.

Deng, J.; Dong, W.; Socher, R.; Li, L.-J.; et al. 2009. Imagenet: A large-scale hierarchical image database. In *CVPR*.

Dietterich, T. G. 2000. Ensemble methods in machine learning. In *International workshop on multiple classifier systems*, 1–15.

Gastaldi, X. 2017. Shake-shake regularization. *arXiv preprint arXiv:1705.07485*.

Goodfellow, I. J.; Warde-Farley, D.; Mirza, M.; Courville, A.; and Bengio, Y. 2013. Maxout networks. In *ICML*.

Goodfellow, I.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A.; and Bengio, Y. 2014. Generative adversarial nets. In *NIPS*.

Hansen, L. K., and Salamon, P. 1990. Neural network ensembles. *IEEE transactions on pattern analysis and machine intelligence* 12(10):993–1001.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep residual learning for image recognition. In *CVPR*.

Hinton, G.; Vinyals, O.; and Dean, J. 2015. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*.

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

Huang, G.; Sun, Y.; Liu, Z.; Sedra, D.; and Weinberger, K. Q. 2016. Deep networks with stochastic depth. In *ECCV*.

Huang, G.; Li, Y.; Pleiss, G.; Liu, Z.; Hopcroft, J. E.; and Weinberger, K. Q. 2017a. Snapshot ensembles: Train 1, get m for free. In *ICLR*.

Huang, G.; Liu, Z.; Weinberger, K. Q.; and van der Maaten, L. 2017b. Densely connected convolutional networks. In *CVPR*.

Ioffe, S., and Szegedy, C. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*.

Johnson, J.; Gupta, A.; and Fei-Fei, L. 2018. Image generation from scene graphs. In *CVPR*.

Krizhevsky, A.; Sutskever, I.; and Hinton, G. 2012. Imagenet classification with deep convolutional neural networks. In *NIPS*.

Krizhevsky, A. 2009. Learning multiple layers of features from tiny images. Technical report.

Krogh, A., and Vedelsby, J. 1995. Neural network ensembles, cross validation, and active learning. In *NIPS*.

Lakshminarayanan, B.; Pritzel, A.; and Blundell, C. 2017. Simple and scalable predictive uncertainty estimation using deep ensembles. In *NIPS*.

Larsson, G.; Maire, M.; and Shakhnarovich, G. 2016. Fractalnet: Ultra-deep neural networks without residuals. *arXiv preprint arXiv:1605.07648*.

Lee, C.-Y.; Xie, S.; Gallagher, P. W.; et al. 2015. Deeply-supervised nets. In *AISTATS*.

Li, Y.; Yang, J.; Song, Y.; Cao, L.; Luo, J.; and Li, L.-J. 2017. Learning from noisy labels with distillation. In *ICCV*.

Liu, Z.; Li, J.; Shen, Z.; Huang, G.; Yan, S.; and Zhang, C. 2017. Learning efficient convolutional networks through network slimming. In *ICCV*.

Maaten, L. v. d., and Hinton, G. 2008. Visualizing data using t-sne. *Journal of machine learning research* 9(Nov):2579–2605.

Netzer, Y.; Wang, T.; Coates, A.; Bissacco, A.; Wu, B.; and Ng, A. Y. 2011. Reading digits in natural images with unsupervised feature learning. In *NIPS workshop on deep learning and unsupervised feature learning*, volume 2011, 5.

Papernot, N.; Abadi, M.; Erlingsson, U.; Goodfellow, I.; and Talwar, K. 2017. Semi-supervised knowledge transfer for deep learning from private training data. In *ICLR*.

Paszke, A.; Gross, S.; Chintala, S.; Chanan, G.; Yang, E.; DeVito, Z.; Lin, Z.; Desmaison, A.; Antiga, L.; and Lerer, A. 2017. Automatic differentiation in pytorch.

Perrone, M. P., and Cooper, L. N. 1995. 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. 342–358.

Romero, A.; Ballas, N.; Kahou, S. E.; Chassang, A.; Gatta, C.; and Bengio, Y. 2015. Fitnets: Hints for thin deep nets. In *ICLR*.

Simonyan, K., and Zisserman, A. 2015. Very deep convolutional networks for large-scale image recognition. In *ICLR*.

Singh, S.; Hoiem, D.; and Forsyth, D. 2016. Swapout: Learning an ensemble of deep architectures. In *Advances in neural information processing systems*, 28–36.

Srivastava, N.; Hinton, G. E.; Krizhevsky, A.; et al. 2014. Dropout: a simple way to prevent neural networks from overfitting. *JMLR*.

Szegedy, C.; Liu, W.; Jia, Y.; Sermanet, P.; et al. 2015. Going deeper with convolutions. In *CVPR*.

Wan, L.; Zeiler, M.; Zhang, S.; Le Cun, Y.; and Fergus, R. 2013. Regularization of neural networks using dropconnect. In *ICML*.

Yim, J.; Joo, D.; Bae, J.; and Kim, J. 2017. A gift from knowledge distillation: Fast optimization, network minimization and transfer learning. In *CVPR*.