Embedding Complementary Deep Networks for Image Classification

Qiuyu Chen[1*], Wei Zhang[2*], Jun Yu[3], Jianping Fan[1]
[1]Department of Computer Science, University of North Carolina at Charlotte, USA
[2]Shanghai Key Laboratory of Intelligent Information Processing
[2]School of Computer Science, Fudan University, China
[3]School of Computer Science and Technology, Hangzhou Dianzi University, China
{qchen12,jfan}@uncc.edu, weizh@fudan.edu.cn, yujun@hdu.edu.cn

Abstract

In this paper, a deep embedding algorithm is developed to achieve higher accuracy rates on large-scale image classification. By adapting the importance of the object classes to their error rates, our deep embedding algorithm can train multiple complementary deep networks sequentially, where each of them focuses on achieving higher accuracy rates for different subsets of object classes in an easy-to-hard way. By integrating such complementary deep networks to generate an ensemble network, our deep embedding algorithm can improve the accuracy rates for the hard object classes (which initially have higher error rates) at certain degrees while effectively preserving high accuracy rates for the easy object classes. Our deep embedding algorithm has achieved higher overall accuracy rates on large-scale image classification.

1. Introduction

With the availability of massive training images and the rapid growth of computational powers of GPUs, we are now able to develop scalable learning algorithms to support large-scale image classification, and deep learning [17, 13, 5, 30, 31, 22, 10, 12] has demonstrated its outstanding performance because it can learn more discriminative representations in an end-to-end fashion. On the other hand, boosting has demonstrated its strong capability by embedding multiple complementary weak classifiers to construct an ensemble one [26, 8, 34]. By assigning larger weights (importance) to hard samples (which are misclassified by the previous weak classifier), boosting can learn a complementary weak classifier at the current training round by paying more attention on such hard samples. Thus it is very attractive to invest whether boosting [26, 8, 34] can be integrated with deep learning to achieve higher accuracy rates on large-scale image classification.

By using deep networks to replace the weak classifiers in traditional boosting frameworks, boosting of deep networks has recently received enough attention and some interesting researches have been done [23, 28, 29, 36, 2, 18, 24, 6, 32]. All these existing deep boosting algorithms simply use the weighted errors (proposed by Adaboost [26, 8, 34]) to replace the softmax errors (used in deep learning), and the underlying deep networks treat the errors from the hard object classes and the easy ones to be equally important. It is worth noting that object classes may have significant differences on their learning complexities (i.e., some object classes could be harder to be recognized than others), thus the errors from the hard object classes and the easy ones may have significantly different effects on optimizing their joint objective function. Therefore, learning a joint deep network for the hard object classes and the easy ones may not be an optimal solution for large-scale visual recognition because such joint deep network may not have strong discrimination ability on the hard object classes which may result in low accuracy rates.

To achieve higher accuracy rates on large-scale image classification, three types of solutions can be invested to diversify the deep networks being combined and generate more discriminative ensemble one: (a) *weightingthe training samples* according to their error rates and most existing deep boosting algorithms [23, 28, 29, 36, 2, 18, 24, 6, 32] belong to this direction; (b) *learning multiple deep networks by using different model parameters or using various sample subsets* and some deep embedding algorithms [15, 25, 33, 9, 27, 14, 20, 4, 21, 1] belong to this direction; (c) *training multiple complementary deep networks*, e.g., such complementary deep networks are trained sequentially by focusing on achieving higher accuracy rates for different subsets of object classes in an easy-to-hard way, so that they can enhance each other. According to the best of our knowledge, the third direction (i.e., training and embedding complementary deep networks) has not been explored so far.

*The first two authors have equal contributions on this work.

Based on these observations, in this paper, a deep embedding algorithm is developed to train and combine multiple complementary deep networks to generate more discriminative ensemble network for achieving higher accuracy rates on large-scale image classification. The rest of the paper is organized as: Section 2 briefly reviews the related work; Section 3 introduces our deep embedding algorithm; Section 4 reports our experimental results over three datasets; and we conclude this paper at Section 5.

Our code is public available at `https://github.com/qychen13/DifficultyAwareEmbedding`.

2. Related Work

In this section, we briefly review the most relevant researches on deep learning, deep boosting, and deep embedding.

Deep learning has demonstrated its outstanding abilities on large-scale image classification [17, 13, 5, 30, 31, 22, 10, 12], but most existing approaches completely ignore that object classes may have significant differences on their learning complexities, e.g., some object classes may be harder to be recognized than others and the gradients of their joint objective function are not uniform for all of them. As a result, learning a joint deep network for the hard object classes and the easy ones may not be an optimal solution for large-scale image classification. Thus it is very attractive to develop new approaches that can learn the deep networks for the easy object classes and the hard ones sequentially in an easy-to-hard way.

By assigning different weights (importance) to the training samples adaptively, boosting [26, 8, 34] has provided an easy-to-hard approach to train multiple complementary weak classifiers iteratively. Some deep boosting algorithms have been developed by seamlessly integrating boosting with deep networks [23, 28, 29, 36, 2, 18, 24, 6, 32]. All these existing deep boosting algorithms use the weighted errors to replace the softmax errors in traditional deep learning frameworks. Because the errors from all the object classes (hard object classes and easy ones) are treated to be equally important, such deep boosting algorithms may still result in low accuracy rates for the hard object classes.

Some deep embedding algorithms have been developed [15, 25, 33, 9, 27, 14, 20, 4, 21, 1], where various sample sets or different model parameters are used to diversify the deep networks but the errors from the hard object classes and the easy ones are treated to be equally important. On the other hand, our deep embedding algorithm focuses on combining multiple complementary deep networks to generate an ensemble network: (a) the errors from the hard object classes are assigned with larger importance; (b) multiple complementary deep networks are trained sequentially and each of them focuses on achieving higher accuracy rates for different subsets of object classes in an easy-to-hard way, so that they can enhance each other.

Algorithm 1 Deep Embedding of Complementary Networks

Require: Training set for N classes: $\{(x_i, y_i) \mid y_i \in \{C_1, ..., C_N\}, i = 1, ..., R\}$; Initial importance for N classes: $\phi_1(C_1) = ... = \phi_1(C_N) = \frac{1}{N}$; Number of complementary deep networks or iterations: τ.

1: **for** $t = 1, \ldots, \tau$ **do**
2: Normalizing: $\varphi_t(C_l) = \frac{\phi_t(C_l)}{\sum_{j=1}^{N} \phi_t(C_j)}$
3: Training the t^{th} deep network $f_t(x)$;
4: Calculating the error rate $\varepsilon_t(C_l)$ for each class;
5: Computing the weighted error rate for $f_t(x)$: $\varepsilon_t = \sum_{l=1}^{N} \varphi_t(C_l)\, \varepsilon_t(C_l)$;
6: Setting $\gamma_t = \frac{\mu\varepsilon_t}{1-\mu\varepsilon_t}$;
7: Updating $\phi_{t+1}(C_l) = \phi_t(C_l)\gamma_t^{1-\mu\varepsilon_t(C_l)}$;
8: **end for**
9: Output: $\mathbb{F}(x) = \frac{1}{\mathbb{Z}} \sum_{t=1}^{\tau} \log\left(\frac{1}{\gamma_t}\right) f_t(x)$

3. Embedding Complementary Deep Networks

As illustrated in Algorithm 1, our deep embedding algorithm contains the following key components: (a) Training the current t^{th} deep network $f_t(x)$ by focusing on achieving higher accuracy rates for the hard object classes which have higher error rates with the previous $(t-1)^{th}$ deep network $f_{t-1}(x)$; (b) Estimating the weighted error rate for $f_t(x)$ according to the distribution of importance for N object classes; (c) Updating the distribution of importance for N object classes according to their error rates by $f_t(x)$, so that the $(t+1)^{th}$ deep network $f_{t+1}(x)$ can spend more efforts on the hard object classes at the next training round; (d) Such iterative process stops when the maximum number of iterations is reached or a certain level of accuracy rates is achieved. Our deep embedding algorithm uses the deep CNNs as its weak learners and many well-designed deep networks can be used.

3.1. Learning Complementary Deep Networks

To train the current t^{th} deep network $f_t(x)$, a deep CNNs is employed to obtain more discriminative representation for the image x, followed by a fully connected discriminant layer and a N-way softmax layer. The output of the t^{th} deep network $f_t(x)$ is a distribution of the prediction probabilities for N object classes, denoted as $f_t(x; \theta_t) = [p_t(C_1|x), ..., p_t(C_N|x)]^\top$, where the lth probability score $p_t(C_l|x)$ is for the image x to be assigned into the lth object class C_l, and θ_t is the model parameter set for the t^{th} deep network $f_t(x)$. Ideally, the t^{th} deep network $f_t(x)$ assigns the image x into the object class with the maximum probability score:

$$\hat{y}^t = \arg\max_l p_t(C_l|x), \quad l \in \{1, \cdots, N\} \tag{1}$$

The training set for N object classes is denoted as: $\{(x_i, y_i) \mid y_i \in \{C_1, ..., C_N\}, i = 1, ..., R\}$, where R is the number of training samples. To train the t^{th} deep network $f_t(x)$, its model parameters can be obtained by maximizing the objective function:

$$\Omega_t(\theta_t) = \sum_{l=1}^{N} \varphi_t(C_l)\xi_{lt} \tag{2}$$

where $\varphi_t(C_l) = \frac{\phi_t(C_l)}{\sum_{j=1}^{N}\phi_t(C_j)}$ is the normalized importance for the lth object class C_l, while $\phi_t(C_l)$ is the unnormalized importance. ξ_{lt} is used to measure the margin between the average confidences for the correctly-classified images and the misclassified images for the lth object class C_l:

$$\begin{aligned}\xi_{lt} = \frac{1}{R_l}\sum_{i=1}^{R}\mathbf{1}(y_i = C_l)\log p_t(C_l|x_i) - \\ \frac{1}{R - R_l}\sum_{i=1}^{R}\mathbf{1}(y_i \neq C_l)\log p_t(C_l|x_i)\end{aligned} \tag{3}$$

where R_l is the number of training images from the lth object class, and $\sum_{l=1}^{N} R_l = R$. The indication function $\mathbf{1}(y_i = C_l)$ is equal to 1 if $y_i = C_l$; otherwise zero. If the second item in Eq.(3) is small enough and negligible, it approximates $\xi_{lt} \approx \frac{1}{R_l}\sum_{i=1}^{R}\mathbf{1}(y_i = C_l)\log p_t(C_l|x_i)$, then maximizing the objective function in Eq.(2) is equivalent to maximizing the weighted likelihood. By using the normalized importance $[\varphi_t(C_1),...,\varphi_t(C_N)]$ to estimate the learning complexities for N object classes, our deep embedding algorithm can push the current deep network $f_t(x)$ to focus on distinguishing the hard object classes which have higher error rates and tend to be misclassified by the previous deep network $f_{t-1}(x)$, thus it can support an easy-to-hard solution for large-scale image classification.

For the t^{th} deep network $f_t(x)$, the error rate $\epsilon_t(C_l)$ for the lth object class C_l is defined as:

$$\begin{aligned}\epsilon_t(C_l) = \frac{1}{2}\sum_{i=1}^{R}\Bigg\{\mathbf{1}(y_i = C_l)\frac{1 - p_t(C_l|x_i)}{R_l} + \\ \mathbf{1}(y_i \neq C_l)\frac{p_t(C_l|x_i)}{R - R_l}\Bigg\}\end{aligned} \tag{4}$$

The error rate in Eq.(4) is calculated in a soft decision way with probability; alternatively, we can also simply compute the error rate in a hard decision way as $\epsilon_t(C_l) = \frac{1}{R}\sum_{i=1}^{R}\mathbf{1}(y_i = C_l \wedge \hat{y}_i^t \neq C_l)$.

The error rate ε_t for $f_t(x)$ is defined as:

$$\varepsilon_t = \sum_{l=1}^{N}\varphi_t(C_l)\epsilon_t(C_l) \tag{5}$$

The importance over N object classes are initialized to be equal: $\phi_1(C_l) = \frac{1}{N}$, $l = 1, ..., N$, and they are updated iteratively according to the error rates:

$$\phi_{t+1}(C_l) = \phi_t(C_l)\gamma_t^{1-\mu\epsilon_t(C_l)} \tag{6}$$

and the normalization:

$$\varphi_{t+1}(C_l) = \frac{\phi_{t+1}(C_l)}{\sum_{j=1}^{N}\phi_{t+1}(C_j)} \tag{7}$$

where μ in Eq.(6) is a hyper-parameter to be selected. γ_t in Eq.(6) is an increasing function of ε_t and its range is $0 < \gamma_t < 1$, and as in Section 3.3, the optimal γ_t in Eq.(6) is set to be:

$$\gamma_t = \frac{\mu\varepsilon_t}{1 - \mu\varepsilon_t} \tag{8}$$

Thus updating the importance for N object classes according to their error rates can push the $(t+1)^{th}$ deep network $f_{t+1}(x)$ to pay more attention on the hard object classes (with larger error rates) at the next training round, so that such sequential deep networks $f_{t+1}(x)$ and $f_t(x)$ are complementary to each other.

3.2. Deep Embedding of Complementary Networks

After τ iterations, we can obtain τ complementary deep networks $\{f_1, \cdots, f_t, \cdots, f_\tau\}$, where each of them focuses on achieving higher accuracy rates on different subsets of N object classes in an easy-to-hard way and they can enhance each other. For recognizing N object classes accurately, all these τ complementary deep networks are embedded to generate an ensemble network $\mathbb{F}(x)$:

$$\mathbb{F}(x) = \frac{1}{\mathbb{Z}}\sum_{t=1}^{\tau}\log\left(\frac{1}{\gamma_t}\right)f_t(x) \tag{9}$$

where $\mathbb{Z} = \sum_{t=1}^{\tau}\log\left(\frac{1}{\gamma_t}\right)$ is a normalization factor. The output of $\mathbb{F}(x)$ is an $N-$dim vector for prediction probability distribution. For a given test sample x_{test}, its lth probability score $p(C_l|x_{test})$ (for assigning it into the lth class C_l) can be easily calculated by Eq.(9).

The given test sample x_{test} is finally be assigned into top-1 object class with the maximum probability score or top-k object classes with top-k scores. By training and combining multiple complementary deep networks, our deep embedding algorithm can generate more discriminative ensemble network $\mathbb{F}(x)$ to achieve higher overall accuracy rates on large-scale visual recognition, e.g., our deep embedding algorithm can improve the accuracy rates for the hard object classes at certain degrees while effectively preserving high accuracy rates for the easy ones.

3.3. Parameter Selection for Deep Embedding

Inspired by [7], we study the optimal parameter selection for deep embedding. In the proposed algorithm, γ_t in the range $[0, 1]$ is set to be an increasing function of the error rate ε_t. γ_t is employed in two folds: (i) As defined in Eq.(6), γ_t is used to update the distribution of importance to pay more attention on the hard object classes with higher error rates; (ii) As defined in Eq.(9), the reciprocal of γ_t is used to determine the weight or importance of the t^{th} complementary deep network $f_t(x)$ in the ensemble network.

The error rate is used as the criterion for the t^{th} deep network $f_t(x)$ to determine the hard object class: $\epsilon_t(C_l) > \frac{1}{2\mu}$, $l \in \{1, \cdots, N\}$, e.g., the error rate for the hard object class is above a threshold $\frac{1}{2\mu}$. For the lth object class, by assessing it over τ complementary deep networks, we can further define $\epsilon_{min}(C_l)$ as:

$$\epsilon_{min}(C_l) \triangleq \min_{t \in \{1, \cdots, \tau\}} \{\epsilon_t(C_l)\}$$

If $\epsilon_{min}(C_l) > \frac{1}{2\mu}$, the lth object class C_l is always hard to be recognized by all τ complementary deep networks. The occurrence of such always-hard object classes may seriously affect the overall accuracy rates of our deep embedding algorithm on large-scale image classification.

We use ϱ to denote the number of such always-hard object classes:

$$\varrho = \sum_{l=1}^{N} \mathbf{1}\left(\epsilon_{min}(C_l) > \frac{1}{2\mu}\right)$$

where $\mathbf{1}(\epsilon_{min}(C_l) > \frac{1}{2\mu}) = 1$ if $\epsilon_{min}(C_l) > \frac{1}{2\mu}$ is true, otherwise, $\mathbf{1}(\epsilon_{min}(C_l) > \frac{1}{2\mu}) = 0$. To achieve higher overall accuracy rates on large-scale visual recognition, we should select suitable γ_t in Eq.(8) to guarantee that $\rho = \frac{\varrho}{N}$ is minimized (e.g., the number of such always-hard object classes ϱ is minimized).

For $0 < \eta < 1$, we have $x^\eta \leq 1 - (1 - x)\eta$. According to Eq.(6), the importance for C_l is updated as:

$$\phi_{t+1}(C_l) = \phi_t(C_l)\gamma_t^{1-\mu\epsilon_t(C_l)}$$

we can get:

$$\begin{aligned}
\sum_{l=1}^{N} \phi_{t+1}(C_l) &= \sum_{l=1}^{N} \phi_t(C_l)\gamma_t^{1-\mu\epsilon_t(C_l)} \\
&\leq \sum_{l=1}^{N} \phi_t(C_l)(1 - (1 - \gamma_t)(1 - \mu\epsilon_t(C_l))) \\
&= \sum_{l=1}^{N} \phi_t(C_l)(1 - (1 - \gamma_t)) + \mu(1 - \gamma_t)\sum_{l=1}^{N} \phi_t(C_l)\epsilon_t(C_l)
\end{aligned} \tag{10}$$

According to Eq.(5) and Eq.(2), we can get:

$$\begin{aligned}
\sum_{l=1}^{N} \phi_t(C_l)\epsilon_t(C_l) &= \left(\sum_{l=1}^{N} \phi_t(C_l)\right)\varepsilon_t \\
\sum_{l=1}^{N} \phi_{t+1}(C_l) &\leq \sum_{l=1}^{N} \phi_t(C_l)(1 - (1 - \gamma_t))+ \\
&\quad \mu(1 - \gamma_t)\left(\sum_{l=1}^{N} \phi_t(C_l)\right)\varepsilon_t \\
&= \left(\sum_{l=1}^{N} \phi_t(C_l)\right)[1 - (1 - \gamma_t)(1 - \mu\varepsilon_t)]
\end{aligned} \tag{11}$$

Because $\sum_{l=1}^{N} \phi_1(C_l) = 1$, we can have:

$$\begin{aligned}
\sum_{l=1}^{N} \phi_2(C_l) &\leq 1 - (1 - \gamma_1)(1 - \mu\varepsilon_1) \\
\sum_{l=1}^{N} \phi_{T+1}(C_l) &\leq \Pi_{t=1}^{\tau}[1 - (1 - \gamma_t)(1 - \mu\varepsilon_t)]
\end{aligned} \tag{12}$$

By substituting Eq.(6) into Eq.(12), we can get:

$$\begin{aligned}
\Pi_{t=1}^{\tau}[1 - (1 - \gamma_t)(1 - \mu\varepsilon_t)] &\geq \sum_{l=1}^{N} \phi_{t+1}(C_l) \\
&= \sum_{l=1}^{N} \left(\phi_1(C_l)\Pi_{t=1}^{\tau}\gamma_t^{1-\mu\epsilon_t(C_l)}\right) \\
&= \frac{1}{N}\sum_{l=1}^{N} \left(\Pi_{t=1}^{T}\gamma_t^{1-\mu\epsilon_t(C_l)}\right) \\
&\geq \frac{1}{N}\sum_{\epsilon_{min}(C_l) > \frac{1}{2\mu}} \left(\Pi_{t=1}^{\tau}\gamma_t^{1-\mu\epsilon_t(C_l)}\right)
\end{aligned} \tag{13}$$

When $\epsilon_{min}(C_l) > \frac{1}{2\mu}$ holds, it guarantees that $\epsilon_t(C_l) > \frac{1}{2\mu}$ and $1 - \mu\epsilon_t(C_l) < \frac{1}{2}$ for all N object classes. Recall the constraint $0 < \gamma_t < 1$, we can have:

$$\begin{aligned}
&\frac{1}{N}\sum_{\epsilon_{min}(C_l) > \frac{1}{2\mu}} \left(\Pi_{t=1}^{\tau}\gamma_t^{1-\mu\epsilon_t(C_l)}\right) \\
&\geq \frac{1}{N}\sum_{\epsilon_{min}(C_l) > \frac{1}{2\mu}} \left(\Pi_{t=1}^{\tau}\gamma_t^{\frac{1}{2}}\right) = \frac{\varrho}{N}\Pi_{t=1}^{\tau}\gamma_t^{\frac{1}{2}}
\end{aligned} \tag{14}$$

Combining Eq.(13) with Eq.(14), we can get:

$$\begin{aligned}
\rho = \frac{\varrho}{N} &\leq \frac{\Pi_{t=1}^{\tau}[1 - (1 - \gamma_t)(1 - \mu\varepsilon_t)]}{\Pi_{t=1}^{\tau}\gamma_t^{\frac{1}{2}}} \\
&= \Pi_{t=1}^{\tau}\frac{1 - (1 - \gamma_t)(1 - \mu\varepsilon_t)}{\gamma_t^{\frac{1}{2}}}
\end{aligned} \tag{15}$$

To minimize the right-side in Eq.(15), we set its partial derivative over γ_t to be zero, and the optimal γ_t is determined as:

$$\gamma_t = \frac{\mu\varepsilon_t}{1-\mu\varepsilon_t}$$

We substitute $\gamma_t = \frac{\mu\varepsilon_t}{1-\mu\varepsilon_t}$ into Eq.(15), and obtain the upper boundary for ρ as:

$$\rho = \frac{\varrho}{N} \leq 2^{\tau}\Pi_{t=1}^{\tau}\sqrt{\mu\varepsilon_t(1-\mu\varepsilon_t)} \tag{16}$$

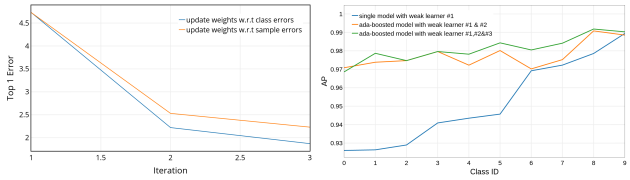
Now we study the range for the hyper-parameter μ. The criterion for the t^{th} deep network $f_t(x)$ to determine the hard object classes (with higher error rates) is defined as $\epsilon_t(C_l) > \frac{1}{2\mu}$, where μ is used to control the threshold of the expected error rate (i.e., when we have more strict requirement on the expected error rate (i.e., smaller threshold), μ should be larger), and we set the constraint for the hyper-parameter μ as $\mu > \frac{1}{2}$. On the other hand, the range of $\gamma_t = \frac{\mu\varepsilon_t}{1-\mu\varepsilon_t}$ is $0 < \gamma_t < 1$, and it is required that $\mu\varepsilon_t < \frac{1}{2}$, i.e., $\mu < \frac{1}{2\varepsilon_t}$. As a result, μ should be selected between the interval $[\frac{1}{2}, \frac{1}{2\varepsilon_t}]$.

From the relationship between $\mu\varepsilon_t$ and $\mu\varepsilon_t(1-\mu\varepsilon_t)$, we can observe the effect of μ on the upper boundary of ρ in Eq.(16): (a) When $\mu \in [\frac{1}{2}, \frac{1}{2\varepsilon_t}]$, i.e., $\frac{\varepsilon_t}{2} < \mu\varepsilon_t < \frac{1}{2}$, the condition $0 < \gamma_t < 1$ is satisfied, and the upper boundary for ρ in Eq.(16) increases when μ increases, the reason is that when μ increases, the threshold for determining the hard object classes is smaller, thus the number of always-hard object classes may increase (i.e., the error rates for more classes could be above such smaller threshold). (b) When $\mu > \frac{1}{2\varepsilon_t}$, i.e., $\mu\varepsilon_t > \frac{1}{2}$. In this case, the condition $0 < \gamma_t = \frac{\mu\varepsilon_t}{1-\mu\varepsilon_t} < 1$ is not satisfied, thus updating the distribution of importance in Eq.(6) can not effectively push the next deep network to pay more attention on the hard object classes. For such situation, large error rates ε_t tend to result in $\mu\varepsilon_t$ being larger than or approaching $\frac{1}{2}$, and γ_t being larger than or approaching 1. Thus the value of μ should be smaller to alleviate large ε_t such that the following constraints can still exist: $\mu\varepsilon_t < \frac{1}{2}$ and $0 < \gamma_t < 1$. (c) When $\mu < \frac{1}{2}$, i.e., $\frac{1}{2\mu} > 1$, it can not be used as the criterion for the t^{th} deep network $f_t(x)$ to determine the hard object classes that satisfy $\epsilon_t(C_l) > \frac{1}{2\mu}$.

4. Experimental Results and Discussions

In this section, we report our evaluation results for our deep embedding algorithm over three popular datasets: MNIST [19], CIFAR-100 [16], and ImageNet1K [3].

(a) Experimental Results on MNIST: MNIST dataset consists of 60,000 training handwritten digit samples and 10,000 test samples [19]. The research in [28] has demonstrated the accuracy improvement on MNIST dataset by updating the sample weights according to their error rates. For fair comparison, we use two approaches to train the deep networks: (1) our deep embedding algorithm updates the class weights according to their error rates; (2) traditional approach updates the sample weights like AdaBoost. In our experiments, we simply train the deep network with the learning rate 0.01 throughout the whole 120 epochs.



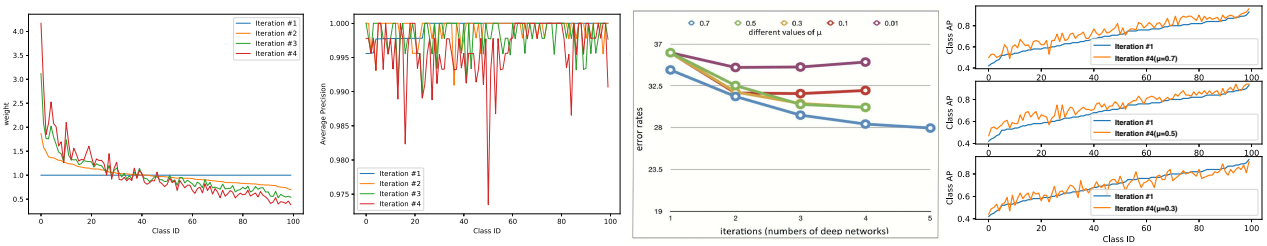
(a) (b)

Figure 1: **Results on MNIST dataset: (a)The comparison on top 1 error for MNIST dataset when different weighting approaches are used; (b)The comparison on AP (average precision) of the ensemble network when different numbers of complementary deep networks are combined.**

With our deep embedding method, the top 1 error rate on the test dataset decreases from 4.73% to 1.87% after three iterations (as shown in Fig. 1a). After the first iteration, the top 1 error of our deep embedding method drops more quickly than the traditional approach. Our deep embedding method, which updates the class weights (i.e., the distribution of importance), can exploit the idea that different classes may have different learning complexities and they should be treated differentially in an easy-to-hard way. From Fig. 1b, one can easily observe that our deep embedding algorithm can significantly improve the accuracy rates for the hard classes while effectively preserving the high accuracy rates for the easy ones.

(b) Experimental Results on CIFAR-100: We also carry out our experiments on CIFAR-100 dataset [16]. CIFAR-100 dataset has 60,000 images for 100 object classes. There are 500 training images and 100 testing images for each class. In the training stage, we hold out 5,000 images for validation and use 45,000 images for training. We further adopt padding, mirroring, shifting for data augmentation and normalization [10, 12]. After several iterations, the training error rate for each class is close to zero [35], thus we update the distribution of importance according to their error rates on the validation datasets. When we train the deep networks on CIFAR-100, the initial learning rate is set to 0.1 and divided by 0.1 at epoch [150,225], and we train the deep networks for 300 epoches. The comparison results are demonstrated in Tab. 1 when (1) different types of deep networks (such as ResNet56($\mu = 0.7$) [11] and DenseNet-BC(k=12) [12]) are used; (2) different numbers T of complementary deep networks are combined.

To train the first deep network, we treat all 100 object classes in CIFAR-100 with equal importance as shown in



(a) (b) (c) (d)

Figure 2: **The comparison on CIFAR-100 dataset: (a) the distribution of weights (importance) for the deep networks at different iterations; (b) the comparison on the accuracy rates for the complementary network on the training set for CIFAR-100 dataset; (c) the effects of using different μ; (d) The comparison on the accuracy rates for the ensemble network on the test set for CIFAR-100 dataset.**

Fig. 2a, and the first deep network is learned and its accuracy rates for all 100 object classes are shown in Fig. 2b. One can easily observe that some easy object classes can achieve acceptable accuracy rates at the first iteration (i.e., the first deep network) but some hard object classes may have very low accuracy rates. By updating the distribution of importance for 100 object classes (putting larger weights for hard object classes and smaller weights for easy object classes) as shown in Fig. 2a, from the second iteration, our deep embedding algorithm can pay more attention on the hard object classes. The effects of the hyper-parameter μ on the performances of our deep embedding algorithm are shown in Fig. 2c.

By combining multiple complementary deep networks, our deep embedding algorithm can generate more discriminative ensemble network to improve the overall accuracy rates on large-scale image classification. As shown in Fig. 2b, one can observe that our complementary deep networks can achieve almost zero error rates on the training image set, which has good correspondence with the observations in [35]. By increasing the importance of hard object classes and pushing the next deep network to pay more attention on them, one can easily observe that the accuracy rates for such hard object classes may improve on the training set (as shown in Fig. 2b), however, the improvement of their accuracy rates on the testing set may still be limited as shown in Fig. 2d and Tab. 1. The reasons for this phenomenon are: (1) such hard object classes may have huge intra-class visual diversities, thus the test images and the training images may have significant differences on their visual properties; (2) such hard object classes may have huge inter-class visual similarities with others, thus they are easily confused from other similar ones. Besides handling the hard object classes and the easy ones sequentially in an easy-to-hard way by weighting their importance according to their error rates, we also need to look for more effective solutions to deal with the issues of huge intra-class visual diversities and huge inter-class visual similarities.

(c) Experimental Results on ImageNet1K: ImageNet1K dataset [3] consists 1,000 object classes, which have 1.2 million images for training, and 50,000 for validation. When we train the deep networks on ImageNet1K dataset, the initial learning rates are set to 0.1 and divided by 0.1 at epoch [30, 60]. The performances of our ensemble network are shown in Tab. 1 when: (1) different types of complementary deep networks are used; (2) different numbers T of complementary deep networks are combined to generate the ensemble network.

When we train the first deep network, we treat all 1,000 object classes with equal importance as shown in Fig. 3(a), and a deep network is learned and its accuracy rates for all 1,000 object classes on the training set are shown in Fig. 3(b). One can easily observe that some easy object classes have achieved acceptable accuracy rates at the first iteration (i.e., by the first deep network) but some hard object classes may have very low accuracy rates. By updating the importance of 1,000 object classes according to their error rates as shown in Fig. 3(a), from the second iteration, our deep embedding algorithm can pay more attention on the hard object classes and their accuracy rates can be improved dramatically on the training set.

Our deep embedding algorithm can generate more discriminative ensemble network to achieve higher accuracy rates on large-scale image classification as shown in Fig. 3(c), e.g., our deep embedding algorithm can improve the accuracy rates for the hard object classes at certain degrees while effectively preserving high accuracy rates for the easy ones. By comparing the performance improvement on the test set (as shown in Fig. 3(c)) and that on the training set (as shown in Fig. 3(b)), we have similar observations as we have obtained on CIFAR-100 dataset: our deep embedding algorithm can improve the accuracy rates for the hard object classes at the training set while the improvement on the testing set may not be so significant, e.g., some hard object classes are always hard for all τ complementary deep networks and ρ is larger for ImageNet1K dataset.

Besides the two reasons (huge intra-class visual diversities and huge inter-class visual similarities) which have discussed above, another key reason for this phenomenon in ImageNet1K dataset is that: (a) Some hard object classes

Table 1: **The comparisons on the top-1 average error rates (%), where the results in () are the top-5 average error rates.**

Datasets	Network	T = 1	T = 2	T = 3	T = 4
MNIST	MLP [19]	4.73	2.22	1.87	1.86
CIFAR-100	ResNet56 [11]	29.53	26.65	24.97	24.15
	DenseNet-BC(k=12) [12]	30.78	28.95	27.60	26.64
ImageNet1K	ResNet50 [11]	24.18(7.49)	23.28(6.98)	22.96(6.81)	22.12(6.79)
	DenseNet121 [12]	25.88(8.38)	24.85(7.89)	23.67(7.25)	22.32(6.17)
	AlexNet [17]	43.71(21.24)	42.61(20.61)	40.83(19.32)	39.23(17.78)

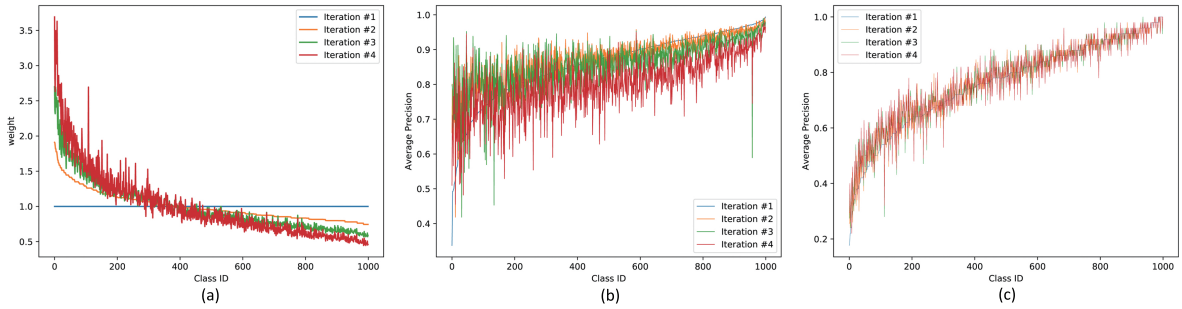


Figure 3: **The comparison on ImageNet1K: (a) the distribution of importance at different iterations; (b) the training accuracy rates for the complementary networks at different iterations; (c) the validation accuracy rates of the embedding networks when different numbers T of complementary networks are trained and combined to form the embedding networks.**

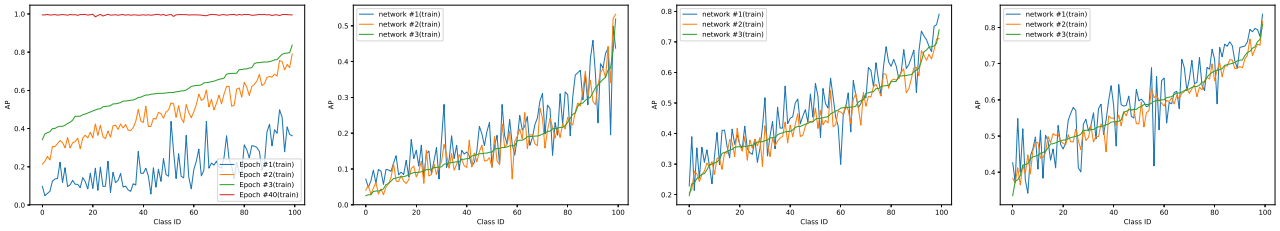
are from the leaf nodes of the concept ontology with longer depths, there may have multiple visually-similar object classes which are hard to be distinguished from each other, e.g., fine-grained hard object classes. As illustrated in Table 1, the top-1 accuracy rates could be very low but the top-5 accuracy rates could be much better because the mistakes on distinguishing among multiple fine-grained hard object classes are not counted in such top-5 error rates. (2) Some hard object classes are from the leaf nodes of the concept ontology with very short depths (i.e., coarse-grained hard object classes), we may need larger numbers of training images to learn the deep networks for discriminating such coarse-grained hard object classes effectively, as a result, using the same number of training images for all the object classes (as it has been done by most existing deep learning algorithms) may not be sufficient to learn discriminative deep networks for such coarse-grained hard object classes.

To learn more discriminative deep networks for the hard object classes, we may further invest: (1) Integrating additional information (such as the inter-class semantic correlations from the concept ontology [3]) to learn the deep network for the fine-grained hard object classes; (2) Using more training images for the coarse-grained hard object classes and developing new deep learning algorithms which can handle sample imbalances effectively; (3) Using heterogeneous embedding, e.g., using different types of deep networks at different iterations such as AlexNet for the first one, ResNet50 for the second complementary one, and ResNet152 for the third complementary one, et al..

(d) Comparison over Embedding Approaches: For CIFAR-100 and ImageNet1K datasets, we have compared three embedding approaches: (1) deep boosting [28, 29]; (2) traditional deep embedding [9, 27]; (3) our deep embedding algorithm. In this comparison experiment, the same type of deep networks (ResNet56 for CIFAR-100 and ResNet50 for ImageNet1k) is used as the complementary network-for three approaches. By combining the same numbers T of deep networks, we have compared the performances of the ensemble networks which are generated by three approaches. As shown in Table 2, one can easily observe that our deep embedding algorithm can achieve higher overall accuracy rates on large-scale image classification.

Further Discussion

(a) Regularization: AdaBoost [7](weighting at the sample level) has achieved higher training accuracy by combining traditional weak learning models (i.e. small networks in [28, 29]), but it is unsuitable for combining large networks: (a) In deep learning, the training error could approach zero and easily suffers from overfitting [35]; (b) When the training error rates are close to zeros, focusing on the hard samples could make the problem of overfitting even worse because the deep networks will be trained with a small portion



(a) network#1 (b) epoch#1 (c) epoch#2 (d) epoch#3

Figure 4: **Illustration for training convergence process of complementary networks dynamically/jointly optimized on CIFAR100: the convergence process of one complementary network(a), the average precision among different networks for the first three epochs(b,c,d). It shows that the difficult categories (small class ID) are similar even with the randomization from the network initialization and optimization process.**

Table 2: **The comparisons on the top-1 average error rates (%) for multiple embedding approaches.**

Datasets	Different Approaches for Network Embedding	T = 1	T = 2	T = 3	T = 4
CIFAR-100	Our deep embedding	29.53	26.65	24.97	24.15
	traditional deep embedding [9, 27]	29.53	29.40	29.35	29.32
	deep boosting [28, 29]	29.53	27.60	26.82	26.53
ImageNet1K	Our deep embedding	24.18	23.28	22.96	22.12
	traditional deep embedding [9, 27]	24.18	23.65	23.15	23.08
	deep boosting [28, 29]	24.18	24.07	23.98	23.75

of samples. For example, in our experiments on CIFAR100 dataset, the training errors could easily approach zeros (Fig. 2a), which is in line with [35], thus we use the validation error rates to weight the objective function in the subsequent iterations. At this scenario, weighting at the sample level is not practical to use the validation results and would only leave a small portion of the samples focused and may worsen the overfitting.

Overall, weighting at the sample level may impair the generation ability of strong complementary learning models (ResNet56/ResNet50 in Tab. 2). Weighting at the category level provides an effective alternative which acts as a regularization method and guides the optimization process (SGD) to pay more attention on difficult categories by optimizing the weighted objective function subsequently (Fig. 2 and Fig. 2d).

(b) Sequentially Guided Optimization: Combining multiple complementary deep networks dynamically/jointly [9, 27] is indeed a reasonable alternative for embedding and it has shown improved results on fine-grained tasks [9] and small neural networks [27], but the improved margin is small with dynamic weighting on non-fine-grained task (CIFAR-100/ImageNet) with deep neural networks (ResNet56/ResNet50) in our experiments (Tab.2). The diversity of dynamic weighting is based on randomization from network initialization and optimization process (SGD) while our proposed method is based on learning difficulty of categories. For difficult categories which are hard to learn and converge slowly for all networks, i.e. small ID categories in Fig.4a for all complementary networks (Fig.4b,4c,4d), the weights/occupations could be distributed almost equally for the complementary networks if optimized jointly and behaves like average ensemble. However, such hard categories would be focused and learned by guided objective function in the subsequent networks with the proposed method (Fig.2&Fig.3). In addition, we have proven the theoretical convergence of our proposed method (Sec. 3.3&Eq. 16).

5. Conclusions

A deep embedding algorithm is developed to train and combine multiple complementary deep networks, where each of them focuses on achieving higher accuracy rates for different subsets of object classes in an easy-to-hard way and they can enhance each other. Our deep embedding algorithm can improve the accuracy rates for the hard object classes at certain degrees while effectively preserving high accuracy rates for the easy ones, thus it can achieve higher overall accuracy rates on large-scale image classification.

Acknowledgment

We would like to thank the anonymous reviewers for their helpful comments. This work was supported in part by NSFC under Grant (No.61473091 and No.61572138) and STCSM Project under Grant No.16JC1420400.

References

[1] Nadav Cohen, Ronen Tamari, and Amnon Shashua. Boosting dilated convolutional networks with mixed tensor decompositions. In *6th International Conference on Learning Representations, ICLR*, 2018.

[2] Corinna Cortes, Mehryar Mohri, and Umar Syed. Deep boosting. In *International Conference on Machine Learning*, pages 1179–1187, 2014.

[3] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[4] Li Deng, Dong Yu, and John Platt. Scalable stacking and learning for building deep architectures. In *2012 IEEE International conference on Acoustics, speech and signal processing (ICASSP)*, pages 2133–2136. IEEE, 2012.

[5] Jeff Donahue, Yangqing Jia, Oriol Vinyals, Judy Hoffman, Ning Zhang, Eric Tzeng, and Trevor Darrell. Decaf: A deep convolutional activation feature for generic visual recognition. In *International conference on machine learning*, pages 647–655, 2014.

[6] Harris Drucker, Robert Schapire, and Patrice Simard. Improving performance in neural networks using a boosting algorithm. In *Advances in neural information processing systems*, pages 42–49, 1993.

[7] Yoav Freund and Robert E Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences*, 55(1):119–139, 1997.

[8] Jerome Friedman, Trevor Hastie, Robert Tibshirani, et al. Additive logistic regression: a statistical view of boosting (with discussion and a rejoinder by the authors). *The annals of statistics*, 28(2):337–407, 2000.

[9] ZongYuan Ge, Alex Bewley, Christopher McCool, Peter Corke, Ben Upcroft, and Conrad Sanderson. Fine-grained classification via mixture of deep convolutional neural networks. In *2016 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 1–6. IEEE, 2016.

[10] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015.

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

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

[13] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional architecture for fast feature embedding. In *Proceedings of the 22nd ACM international conference on Multimedia*, pages 675–678. ACM, 2014.

[14] Tae-Kyun Kim, Ignas Budvytis, and Roberto Cipolla. Making a shallow network deep: Conversion of a boosting classifier into a decision tree by boolean optimisation. *International journal of computer vision*, 100(2):203–215, 2012.

[15] Peter Kontschieder, Madalina Fiterau, Antonio Criminisi, and Samuel Rota Bulo. Deep neural decision forests. In *Proceedings of the IEEE international conference on computer vision*, pages 1467–1475, 2015.

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

[17] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[18] Vitaly Kuznetsov, Mehryar Mohri, and Umar Syed. Multi-class deep boosting. In *Advances in Neural Information Processing Systems*, pages 2501–2509, 2014.

[19] Yann LeCun, Léon Bottou, Yoshua Bengio, Patrick Haffner, et al. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[20] Chen-Yu Lee, Patrick W Gallagher, and Zhuowen Tu. Generalizing pooling functions in convolutional neural networks: Mixed, gated, and tree. In *Artificial intelligence and statistics*, pages 464–472, 2016.

[21] Jun Li, Heyou Chang, and Jian Yang. Sparse deep stacking network for image classification. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015.

[22] Min Lin, Qiang Chen, and Shuicheng Yan. Network in network. In *International Conference on Learning Representations, ICLR*, 2014.

[23] Mohammad Moghimi, Serge J Belongie, Mohammad J Saberian, Jian Yang, Nuno Vasconcelos, and Li-Jia Li. Boosted convolutional neural networks. In *BMVC*, pages 24–1, 2016.

[24] Zhanglin Peng, Ya Li, Zhaoquan Cai, and Liang Lin. Deep boosting: joint feature selection and analysis dictionary learning in hierarchy. *Neurocomputing*, 178:36–45, 2016.

[25] Samuel Rota Bulo and Peter Kontschieder. Neural decision forests for semantic image labelling. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 81–88, 2014.

[26] Robert E Schapire and Yoram Singer. Improved boosting algorithms using confidence-rated predictions. *Machine learning*, 37(3):297–336, 1999.

[27] Jurgen Schmidhuber. Multi-column deep neural networks for image classification. In *Proceedings of the 2012 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3642–3649. IEEE Computer Society, 2012.

[28] Holger Schwenk and Yoshua Bengio. Adaboosting neural networks: Application to on-line character recognition. In *International Conference on Artificial Neural Networks*, pages 967–972. Springer, 1997.

[29] Holger Schwenk and Yoshua Bengio. Boosting neural networks. *Neural computation*, 12(8):1869–1887, 2000.

[30] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *In-*

ternational Conference on Learning Representations, ICLR, 2015.

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

[32] Ilya O Tolstikhin, Sylvain Gelly, Olivier Bousquet, Carl-Johann Simon-Gabriel, and Bernhard Schölkopf. Adagan: Boosting generative models. In *Advances in Neural Information Processing Systems*, pages 5424–5433, 2017.

[33] Andreas Veit, Michael J Wilber, and Serge Belongie. Residual networks behave like ensembles of relatively shallow networks. In *Advances in neural information processing systems*, pages 550–558, 2016.

[34] Paul Viola and Michael J Jones. Robust real-time face detection. *International journal of computer vision*, 57(2):137–154, 2004.

[35] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. In *5th International Conference on Learning Representations, ICLR*, 2017.

[36] Ligang Zhou and Kin Keung Lai. Adaboosting neural networks for credit scoring. In *The Sixth International Symposium on Neural Networks (ISNN 2009)*, pages 875–884. Springer, 2009.