HHS Public Access

Author manuscript

J Appl Stat. Author manuscript; available in PMC 2019 October 19.

Published in final edited form as:

J Appl Stat. 2018 ; 45(15): 2800–2818. doi:10.1080/02664763.2018.1441383.

The Relative Performance of Ensemble Methods with Deep Convolutional Neural Networks for Image Classification

Cheng Ju[a], **Aurélien Bibaut**[a], **Mark van der Laan**[a]

[a] University of California, Berkeley

Abstract

Artificial neural networks have been successfully applied to a variety of machine learning tasks, including image recognition, semantic segmentation, and machine translation. However, few studies fully investigated ensembles of artificial neural networks. In this work, we investigated multiple widely used ensemble methods, including unweighted averaging, majority voting, the Bayes Optimal Classifier, and the (discrete) Super Learner, for image recognition tasks, with deep neural networks as candidate algorithms. We designed several experiments, with the candidate algorithms being the same network structure with different model checkpoints within a single training process, networks with same structure but trained multiple times stochastically, and networks with different structure. In addition, we further studied the over-confidence phenomenon of the neural networks, as well as its impact on the ensemble methods. Across all of our experiments, the Super Learner achieved best performance among all the ensemble methods in this study.

Keywords

Ensemble Learning; Super Learner; Convolutional Neural Network

1. Introduction

Ensemble learning methods train several baseline models, and use some rules to combine them together to make predictions. The ensemble learning methods have gained popularity because of their superior prediction performance in practice. Consider a prediction task with some fixed data generating mechanism. The performance of a particular learner depends on how effective its searching strategy is in approximating the optimal predictor defined by the true data generating distribution, thus it is generally impossible to know a priori which learner would perform best given the finite sample data set and prediction problem [42]. One widely used method is to use cross-validation to give an “objective” and “honest” assessment of each learners, and then select the single algorithm that achieves best validation-performance. This is known as the discrete Super Learner selector [32, 41, 42], which asymptotically performs as well as the best base learner in the library, even as the number of candidates grows polynomial in sample size.

CONTACT Cheng Ju. cju@berkeley.edu.

Instead of selecting one algorithm, another approach to guarantee the predictive performance is to compute the optimal convex combination of the base learners. The idea of ensemble learning, which combines predictors instead of selecting a single predictor, is well studied in the literature: [3] summarized and referred several related studies [1, 10, 15, 33, 34] about the theoretical properties of ensemble learning. Two widely used ensemble techniques are bagging [2] and boosting [11–13]. Bagging uses bootstrap aggregation to reduce the variance for the strong learners, while boosting algorithms "boost" the capacity of the weak learners. [3, 45] proposed a linear combination strategy called stacking to ensemble the models. [42] further extended stacked generalization with a cross-validation based optimization framework called Super Learner, which finds the optimal combination of a collection of prediction algorithms by minimizing the cross-validated risk. Recently, the super learner have showed great success in variety of areas, including precision medicine [27], mortality prediction[5, 31], online learning [**?**], and spatial prediction[8].

In recent years, deep artificial neural networks (ANNs) have led to a series of break-throughs in a variety of tasks. ANNs have shown great success in almost all machine learning related challenges across different areas, like computer vision [17, 23, 40], machine translation [6, 28], and social network analysis [16, 30]. Due to their high capacity/ flexibility, deep neural networks usually have high variance and low bias. In practice, model averaging with multiple stochastically trained networks is commonly used to improve the predictive performance. [23] won the first place in the image classification challenge of ILSVRC 2012, by averaging 7 CNNs with same structure. [36] won the first place in classification and localization challenge in ILSVRC 2014 with averaging of multiple deep CNNs. [17] won the first place using six models of Residual Network with different depth to form an ensemble in ILSVRC 2015. In addition, [17] also won the ImageNet detection task in ILSVRC 2015 with the ensemble of 3 residual network models.

However, the behavior of ensemble learning with deep networks is still not well studied and understood. First, most of the neural networks literature focuses mainly on the design of the network structure, and only applies naive averaging ensemble to enhance the performance. To the best of our knowledge, no detailed work investigates, compares and discusses ensemble methods for deep neural networks. Naive unweighted averaging, which is largely used, is not data-adaptive and thus vulnerable to a "bad" library of base learners: it works well for networks with similar structure and comparable performance, but it is sensitive to the presence of excessively biased base learners. As the deep neural networks are usually sensitive to hyper-parameters and vulnerable to overfitting, it is reasonable to expect that some base learners in library may fail. This issue could be easily addressed by a cross-validation based data-adaptive ensemble like the Bayes Optimal Classifier and the Super Learner. In later sections, we investigate and compare the performance of several commonly used ensemble methods on an image classification task, with deep convolutional neural networks (CNNs) as base learners.

This study mainly focuses on the comparison of ensemble methods of CNNs for image recognition. We use the CIFAR10 dataset [22], a commonly used benchmark dataset, for experiments and prediction accuracy as criterion to evaluate the performance. The dataset has predetermined training and testing set, so there is little concern of the selection bias in

training/testing splitting. For readers who are not familiar with deep learning, each CNN could be just viewed as a black-box estimator, with an image as input, and outputs the probability vector for each possible class. We refer the interested reader to [14, 25] for more details about deep learning.

2. Background

In this paper, “algorithm candidate”, “hypothesis”, and “base learner” refer to an individual learner (here a deep CNN) used in an ensemble. The term ’library’ refers to the set of the base learners for the ensemble methods.

2.1. Unweighted Average

Unweighted averaging is the most common ensemble approach for neural networks. It takes unweighted average of the output score/probability for all the base learners, and reports it as the predicted score/probability.

Due to the high capacity of deep neural networks, simple unweighted averaging improves the performance substantively. Taking the average of multiple networks reduces the variance, as deep ANNs have high variance and low bias. If the models are uncorrelated enough, the variance of models could be dramatically reduced by averaging. This idea inspires Random Forest [4], which builds less correlated trees by bootstrapping observations and sampling features.

We could average either directly the score output, or the predicted probability after softmax transformation:

$$p_{ij} = softmax\left(\overrightarrow{s}_i\right)[j] = \frac{\overrightarrow{s}_i[j]}{\sum_{k=1}^{K} \exp\left(s_i[k]\right)},$$

where score vector $\overrightarrow{s}_i$ is the output from the last layer of the neural network for i-th unit, $\overrightarrow{s}_i[k]$ is the score corresponding to k-th class/label, and p_{ij} is the predicted probability for unit i in class j. It is more reasonable to average after the softmax transformation, as the scores might have varying scales of magnitude across the base learners, as the score output from different network might be in different magnitude. Indeed, adding a constant to scores for all the classes leaves predicted probability unchanged. In this study, we compared both naive averaging of the scores and averaging of their softmax transformed counterparts (i.e. the probabilities)

Unweighted averaging might be a reasonable ensemble for similar base learners of comparable performance, as the deep learning literature suggests [17, 36, 40]. However, when the library contains heterogeneous networks, the naive unweighted averaging may not be a smart choice. It is vulnerable to the weaker learners in the library, and sensitive to the over-confident candidate (We will explain further the over-confidence phenomenon in later sections.). A good meta-learner should be intelligent enough to combine the strength of base learners data-adaptively. Heuristically, some networks might have weak overall prediction

strength, but can be good at discriminating certain subclasses (e.g. fine-grained classifier). We hope the meta-learner could combine the strengths of all the base learners, thus yielding a better strategy.

2.2. Majority Voting

Majority voting is similar to unweighted averaging. But instead of averaging over the output probability, it counts the votes of all the predicted labels from the base learners, and makes a final prediction using label with most votes. Or equivalently, it takes an unweighted average using the label from base learners and chooses the label with the largest value.

Compared to naive averaging, majority voting is less sensitive to the output from a single network. However, it would still be dominated if the library contains multiple similar and dependent base learners. Another weakness of majority voting is the loss of information, as it only uses the predicted label.

[24] showed pairwise dependence plays an an important role in majority voting. For image classification, shallow networks usually give more diverse prediction compared to deeper networks[7]. Thus we hypothesize majority voting would yield a greater improvement over base learners with a library of shallow networks than with a library of deep networks.

2.3. Bayes Optimal Classifier

In a classification problem, it can be shown that the function f of the predictors $\mathbf{x}$ that minimizes the misclassification rate $\mathbb{E}I(f(x) \neq y)$ is the so-called Bayes classifier. It is given by $f(x) = \text{argmax}_y P[y|\mathbf{x}]$. It fully characterized by the data-generating distribution P.

In the Bayesian voting approach, each base learner h_j is viewed as an hypothesis made on the functional form of the conditional distribution of y given $\mathbf{x}$. More formally, denoting S_{train} our training sample, and $(\mathbf{x}, y)$ a new data-point, we denote $h_j(y|\mathbf{x}) = P[y|\mathbf{x}, h_j, S_{train}]$. It means the value of the hypothesis h_j, which is trained on S_{train}, evaluated at (y, x). The Bayesian voting approach requires a prior distribution that, for each j, models the probability $P(h_j)$ that the hypothesis h_j is correct. Using the Bayes rule, one readily obtains that

$$P(y|\mathbf{x}, S_{train}) \propto \sum_{h_j} P[y|h_j, \mathbf{x}, S_{train}]P[S_{train}|h_j]P[h_j]\,. \tag{1}$$

This motivates the definition of the Bayesian Optimal classifier as

$$\text{argmax}_y \sum_{h_j} h_j(y|\mathbf{x}) P[S_{train}|h_j] P[h_j] \,. \tag{2}$$

Note that $P\left[S_{train}\middle|h_j\right] = \prod_{(y,x)\in S_{train}} h_j(y\middle|x)$ is the likelihood of the data under the hypothesis h_j. However this quantity might not reflect well the quality of the hypothesis since the likelihood of the training sample is subject to overfitting. To give an "honest" estimation, we could split the training data into two sets, one for model training, and the other for computing $P[S_{\text{train}}|h]$. For neural networks, a validation set (distinct from the testing set) is usually set aside only to tune a few hyper-parameters, thus the information in it is not fully exploited. We expect that using such a validation set would provide a good estimation of the likelihood $P[S_{train}|h]$. Finally, we would assess the model using the untouched testing set.

The second difficulty in BOC is choosing the prior probability for each hypothesis $p(h_j)$. For simplicity, the prior is usually set to be the uniform distribution [29].

[9] observed that, when the sample size is large, one hypothesis typically tends to have a much larger posterior probability than others. We will see in the later section that when the validation set is large, the posterior weight is usually dominated by only one hypothesis (base learner). As the weights are proportional to the likelihood on the validation set, if the weight vector is dominated dominated by a single algorithm, BOC would be the same selector as the discrete Super Learner selector with negative likelihood loss function [42].

2.4. Stacked Generalization

The idea of stacking was originally proposed in [45], which concludes stacking works by deducing the biases of the generalizer(s) with respect to a provided learning set. [3] also studied stacked regression by using cross-validation to construct the 'good' combination.

Consider a linear stacking for the prediction task. The basic idea of stacking is to 'stack' the predictions $f_1, \cdots, f_m$ by linear combination with weights a_i, $i \in 1, \cdots, m$:

$$f_{stacking}(\mathrm{x}) = \sum_{i=1}^{m} a_i f_i(\mathrm{x})$$

where the weight vector a is learned by a meta-learner.

3. Super Learner: a Cross-validation based Stacking

Super Learner [42] is an extension of stacking. It is a cross-validation based ensemble framework, which minimizes cross-validated risk for the combination. The original paper

[42] demonstrated the finite sample and asymptotic properties of the Super Learner. The literature shows its application to a wide range of topics, e.g. survival analysis [19], clinical trial [37], and mortality prediction [31]. It combines the base learners by cross-validation. Here is an example of SL with V-fold cross-validation with m base learners for binary prediction. We first define the cross-validated loss for j-th base learner:

$$R_{CV}^{(j)} = \sum_{v=1}^{V} \sum_{i \in val(v)} l\left(y_i, p_{ji}^{-v}\right)$$

where $val(v)$ is the set of indices of the observations in the v-th fold, and p_{ji}^{-v} is defined as the prediction for the i-th observation, from the j-th base learner that trained on the whole data except the v-th fold. Then we have

$$R_{CV}(\vec{a}) = \sum_{v=1}^{V} \sum_{i \in val(v)} l\left(y_i, \sum_{j=1}^{m} a_j p_{ji}^{-v}\right)$$

where $\vec{a} = [a_1, \cdots, a_m]$ is the weight vector. The optimal weight vector given by the Super Learner is then

$$\vec{a} = \underset{\vec{a}}{\operatorname{argmin}} R_{CV}(\vec{a})$$

For simplicity, we consider the binary classification task, which could be easily generalized to multi-class classification and regression. We first study a simple version of the Super Learner with m single algorithms, using negative (Bernoulli) log-likelihood as loss function:

$$l(y, p) = -[y\log(p) + (1-y)\log(1-p)].$$

Thus the cross-validated loss is:

$$R_{CV}(\vec{a}) = -\sum_{v=1}^{V} \sum_{i \in val(v)} [y_i \log(\sum_{j=1}^{m} a_j p_{ji}^{-v}) + (1-y_i)\log(1 - \sum_{j=1}^{m} a_j p_{ji}^{-v})]$$

where p_{ji}^{-v} is the predicted probability for i-th unit from j-th base learner which is trained on the whole data except v-th fold.

In addition, stacking on the logit scale usually gives much better performance in practice. In other words, we use the optimal linear combination before softmax transformation:

$$R_{CV}(\vec{a}) = \sum_{v=1}^{V} \sum_{i \in val(v)} l(y_i, expit(\sum_{j=1}^{m} a_j logit(p_{ji}^{-v})))$$

For K-class classification with softmax output like neural networks, we could also ensemble in the score level:

$$p_i^z(\vec{a}) = -\log\left(\frac{\exp\left(\sum_{j=1}^{m} a_j \cdot s_i[j,z]\right)}{\sum_{k=1}^{K}\exp\left(\sum_{j=1}^{m} a_j \cdot s_i[j,k]\right)}\right)$$

where $p_i^z(\vec{a})$ is the ensemble prediction for i-th unit and z-th class with weight vector $\vec{a}$. $\mathbf{s}_i$ is an m by K matrix, and $s_i[j, k]$ stands for the score of j-th model and k-th class.

We can impose restrictions on a, such as constraining it to lie in a probability simplex:

$$\|a\|_1 = 1, a_i \geq 0, \text{ for } i = 1, \cdots, m.$$

This would drive the weights of some base learners to zero, which would reduce the variance of the ensemble and make it more interpretable. This constrain is not a necessary condition to achieve the oracle property for SL. In theory, the oracle inequality requires bounded loss function, so the LASSO constraint is highly advisable (e.g. $\sum_j |a_j| < M$, for some fixed M). In practice, we found imposing large M leads to better practical performance.

For small data sets, it is recommended to use cross-validation to compute the optimal ensemble weight vector. However this takes a long time when the data set and the library are large. Usually people just set aside a validation set, instead of cross-validation, to assess and tune the models for deep learning. Similarly, instead of optimizing the V-fold cross-validated loss, we could optimize on the single-split cross-validation loss instead to get the ensemble weights, which is so called “single split (or sample split) Super Learner”. Figure 1 shows the details of this variation of Super Learner. [21] shows the success of such single split Super Learner in three large healthcare databases. In this study, we compute the weights of the Super Learner by minimizing the single-split cross-validated loss. This procedure necessitates almost no additional computation: only one forward pass for all validation images and then solving a low-dimensional convex optimization.

3.1. Super Learner From a Neural Network Perspective

Lots of neural network structures could be considered as ensemble learning. One of the commonly used regularization methods for deep neural network, dropout [38], randomly removes certain proportion of the activations (the output from the last layer) during the training and uses all the activations in the testing. It could be seen as training multiple base learners and ensemling them during prediction. [43] discusses ResNet, a state-of-the-art network structure, could be understood as an exponential ensembles of shallow networks. However, such ensembles might be highly biased, as the meta-learner computes the weights based on the prediction of the base learner (e.g. shallow network) on the training set. These weights might be biased as the base-learners might not make objective prediction on the training set.

In contrast, the Super Learner computes an honest ensemble weight based on the validation set. A validation set is commonly used to train/tune a neural network. However, it is usually only used to select a few tuning parameters (e.g. learning rate, weight decay). For most image classification data sets, the validation set is very large in order to make the validation stable. We thus conjecture that the potential of the validation information has not been fully exploited.

The Super Learner could be considered as a neural network with 1 by 1 convolution over the validation set, with the scores of the base learners as input. It learns the $1 \times 1 \times m$ kernel either by back-propagation, or through directly solving the convex optimization problem.

4. Experiment

4.1. Data

The CIFAR-10 data set [22] is a widely used benchmark data set for image recognition. It contains 10 classes of natural images, with 50, 000 training images and 10, 000 testing images. Each image is an RGB image of size 32 × 32. There are 10 classes in the data set: airplane, automobile, bird, cat, deer, dog, frog, horse, ship, and truck. Each class has 5000 images in the training data and 1000 images in the testing data.

4.2. Network description

In this section we introduce several popular neural networks as our base learners. For each learner, we used the hyper-parameters used in the paper that originally introduced it.

4.2.1. Network in Network—The network in network (NIN) structure [26] consists of mlpconv (MLP) layers, which use multilayer perceptrons to convolve the input. Each MLP layer is made by one convolution layer with larger kernel size followed by two 1 × 1 convolution layer and max pooling layer. In addition, it uses a global average pooling layer as a replacement for the fully connected layers in conventional neural networks.

4.2.2. GoogLeNet—GoogLeNet [40] is a deep convolutional neural network architecture based on the inception module, which improved the computational efficiency. In each inception module, a 1 × 1 convolution is applied as dimension reduction before expensive large convolutions. Within each inception module, the propagation splits into 4 flows, each with different convolution size, and is then concatenated.

4.2.3. VGG Network—VGG net [36] is a neural network structure using an architecture with very small (3×3) convolution filters, which won the first and the second places in the localization and classification tracks for ImageNet Challenge 2014 respectively. Each block is made by several consecutive 3 × 3 convolutions and followed by a max pooling layer. The number of filters for each convolution increases as the network goes deeper. Finally there are three fully connected layers before the softmax transformation.

In this study, we only used VGG net D with 16 layers [36]. We denote it as VGG net for simplicity in the later sections.

4.2.4. Residual Network—Residual Network [17] is a network structure that stacked by multiple "bottleneck" building blocks. Figure 5 shows an example of so called bottleneck building block, stacked by two regular layer (e.g. convolution layers). In the original study [17], each bottleneck building block is made by three convolutional layers, with kernel size 1, 3, and 1. Similar to NIN and GoogLeNet, it uses 1 × 1 convolution as dimension reduction to reduce the computation. There is a parameter-free identity shortcut from the starting layer to the final output for each bottleneck block. It solves the degradation problem for deep networks and makes training a very deep neural network possible.

In later sections, we follow the same structure from the original paper for CIFAR-10 data: we use a stack of $6n$ layers with 3×3 convolutions. The sizes of the feature maps are {32, 16, 8} respectively, with $2n$ layers for each feature map size [17]. There would be $6n + 2$ layers including the softmax layer. For example, ResNet with $n = 5$ has 32 layers in total.

4.3. Training

For all the models, we split the training data into training (first 4, 5000 images) and validation set (last 5, 000 images). There are 10K testing data.

For the Network-in-Network model, we used Adam with learning rate 0.001. We followed the original paper [26], tuning the learning rate and initialization manually. The training was regularized by L-2 penalty with predefined weight 0.001 and two dropout layers in the middle of the network, with rate 0.5.

For VGG net, we slightly modified the training procedure in the original paper [36] for ILSVRC-2013 competitions [35, 46]. We used SGD with momentum 0.9. We started with learning rate 0.01 and decay divide it by 10 at every $32k$ iterations. The training is regularized by L-2 penalty with weight 10^{-3} and two dropout layers for the fitst two fully connected layer, with rate 0.5.

For GoogLeNet, we set base learning rate to be 0.05, weight decay 10^{-3}, and momentum 0.9. We decreased the learning rate by 4% every 8 epochs. We set the rate to 0.4 for the dropout layer before the last fully connected layer.

For the Residual Network, we follow the training procedures in the original paper [17]: we applied SGD with weight decay of 0.0001 and momentum of 0.9. The weight was initialized following the method in [18], and we applied batch normalization [20] without dropout. Learning rate started with 0.1, and was divided by 10 at every $32k$ iterations. We trained the model with 200 epochs.

All the networks were trained with mini-batch size 128 for 200 epochs.

4.4. Results

In this section, we compare the empirical performance for all the ensemble methods we mentioned before, including: Unweighted Averaging (before/after softmax layer), Majority Voting, Bayes Optimal Classifier, Super Learner (with negative log-likelihood loss). We also include discrete SL, with negative log-likelihood loss and 0–1 error loss.. For comparison,

we list the base learner which achieved best performance on the *testing set*, as an empirical oracle.

4.4.1. Ensemble of Same Network with Different Training Checkpoints—Table 1 shows the prediction accuracy for the ResNet 8 and 110 after different epochs. As ResNe 8 is much shallower, thus more adaptive during training, we set the smaller interval with epoch 10. Notice there is a great accuracy improvement around epoch 100, due to the learning rate decay.

For ResNet 8, the SL is substantively better than naive averaging and majority voting. Earlier stage learners would have worse performance, which causes the deterioration of the performance for naive averaging. The performance of majority voting is even worse than the best base learner, as the majority of the base learners are under-optimized.

For ResNet 110, the performance for all the meta-learners is similar. One possible explanation is that deeper network is more stable during training.

In this experiment, the weights of BOCs are dominated by one model, which gives the best performance on the validation set. Thus the BOC is equivalent to the discrete Super Learner with negative likelihood as loss function. In the experiments, BOC performed only as well as the best base learner. In the subsequent experiments, all the BOCs showed the similar dominated weight pattern. Given the practical equivalence with the discrete Super Learner, we don't elaborate further on BOCs, and we will report only the discrete Super Learner's performance.

4.4.2. Ensemble of Same Network Trained Multiple Times—Unlike other conventional machine learning algorithms, deep neural networks solve a high-dimensional non-convex optimization problem. Mini-batch stochastic gradient descent with momentum is commonly used for training. Due to non-convexity, networks with same structure but different initialization and training vary a lot. [7] studied the distribution of loss on the testing set for a certain network structure trained multiple times with SGD. It shows the distribution of loss is more concentrated for deeper neural network. This suggest deep neural networks are less sensitive to randomness in the initialization and training. If so, ensemble learning would be less helpful for the deeper nets.

To help understand this property, we trained 4 ResNet with 8 layers and 4 ResNet with 110 layers.

We trained 4 networks for ResNet 8 and 110 respectively. Table 3 shows the performance of the networks. We further studied the performance of all the meta-learners. Shallow networks enjoyed more improvement (2.54%) compared to deeper networks 1.43% after ensembled by the Super Learner. Due to the similarity of the models, the SL did not show great improvement compared to naive averaging. Similarly, majority voting did not work well, which might also be due to the diversity of the base learners. The discrete SL with negative log-likelihood loss successfully selected the best single learner in the library, while the discrete SL with error loss selected a slightly weaker one. This suggests that for finite

samples, the Super Learner using the negative log likelihood loss performs better w.r.t. prediction accuracy, than the Super Learner that uses prediction accuracy as criterion.

4.4.3. Ensemble of Networks with Different Structure—In this section, we studied ensemble of networks with different structure. We trained NIN, VGG,and ResNet with 32, 44, 56, 110 layers. Table 5 shows the performance of each net on the testing set.

4.4.4. Over-confident Model—As the $0-1$ loss for classification is not differentiable, cross-entropy loss is commonly used as surrogate loss in neural network training. We could see from table 6 that the cross-entropy is usually negatively correlated with the prediction accuracy. However, we could see that Network-in-Network model has much lower cross-entropy loss compared to all the other models, while it gives worse prediction accuracy. This due to its prediction behavior: we look at the predicted probability of the true labels for the images in the testing set:

It is interesting to observe the high-confidence phenomenon for the Network-in-Network model, where most of the predictions are made with high confidence (predicted probability). Such high-confident networks usually achieve much smaller surrogate loss (negative log-likelihood loss in our example) on the testing set, but not necessary smaller 0–1 error loss. Though all the networks suffered from over-fitting, only the NIN net showed the over-confidence. In addition, NIN has higher training cross-entropy loss (0.13104) compared to VGG (0.02233). Thus it is not reasonable to blindly attribute the over-confidence to the over-fitting.

When several base learners suffer from the over-confidence issue, the performance of model averaging would be seriously deteriorated: the unweighted average score/probability would be dominated by the over-confident models. When all the models are over-confident, the unweighted average is identical to the majority vote.

In addition, the VGG net and the ResNet with 32 layers had very similar predicted probability, even though their structure is totally different (agree on first 3 digits on most observations). However, this special pattern is beyond the scope of this study.

We empirically study the impact of over-confident network candidates for ensemble methods: we have five candidates in the ensemble library: NIN, VGG, ResNet 32, ResNet 44, and ResNet 56. We compare the performance with/without adding NIN, which is the only over-confident net.

Table 8 shows the performance of the ensemble algorithms on the testing set. The unweighted average model was weakened by the NIN net: over-confidence made NIN dominate the others, and led to 0.23% (before softmax) and 5% (after softmax) decrease in the prediction accuracy. The naive average before softmax was less influenced as the scale of networks are different. The majority vote algorithm was not influenced too much by the extra candidate, which is not surprising. The over-confident network only weakened the discrete SL with negative log-likelihood loss, while did not influence the discrete SL with error loss. The Super Learner successfully harnessed the over-confident model: adding NIN helped increase the prediction accuracy from 0.9405 to 0.9414.

4.4.5. Learning from Weak Learner—We hope our ensemble method could learn from all the models, even though there might be base learners with weaker overall performance compared to the other learners in the library. In this experiment, we used under-trained GoogLeNets [40] as the weak candidates. The original paper [40] did not describe explicitly how to automatically train/tune the network in CIFAR 10 data set. We set the initial learning rate to be 0.05, with momentum 0.96, and decreased the learning rate by 4% every 8 epochs. This did not give satisfactory performance: the prediction accuracy on the testing set is around 0.83. To avoid the impact of over-confidence, we removed the NIN net. Thus the weakest base learner in the library is the VGG net, which achieved 0.8914 accuracy on the testing set. We observe that the difference in prediction accuracy for the VGG net and the GoogLeNet is around 6%, which means our GoogLeNet model is substantially weaker than other candidates.

We trained the GoogLeNet 5 times and then compare the performance of different ensemble methods with/without such 5 googLeNets in the library.

In the experiment, adding many weaker candidates deteriorated the performance of the unweighted average. The majority voting was slightly influenced when there were only few weak learners, while would be dominated if the number of the weak learner was large. Unweighted averaging also failed in this case. BOCs remained unchanged as the likelihood on the validation set is still dominated by the same base learner. Super Learner shows exciting success in this setting: the prediction accuracy remained stable with the extra weak learning.

4.4.6. Prediction with All Candidates—As the number of base learners is usually much smaller than the sample size and there is usually no apriori which learner would achieve best performance, it is encouraged to apply as rich library as possible to improve the performance of Super Learner. In this experiment, we simply put all the networks mentioned before into the library of all the ensemble methods.

Table 10 shows the performance of all the ensemble methods as well as the base learner with the best performance. Due to the large proportion of weak learners (e.g. under-fitted GoogLeNet, and the networks trained with less iterations in the first experiment) and the over-confident learners (NIN), all the other ensemble methods have much worse performance compared to Super Learner. This is another strength of the Super Learner: by simply putting all the potential base learners into the library, the Super Learner computes the weights data-adaptively, which does not require any tedious pre-selecting procedure based on human experience.

4.5. Discussion

We studied the relative performance, on the CIFAR 10 dataset, of several widely used ensemble methods, using convolutional neural networks as base networks.

- The unweighted averaging proved surprisingly successful when the performance of the base learners is comparable. It outperformed majority voting in almost all the experiments. However, in section 4.4.4 we found that the unweighted

averaging proved to be sensitive to over-confident candidates. The Super Learner adresses the over confidence issue by optimizing learners's weights on a validation set.

- In section 4.4.1, we observe that ensembling several instances of the same model with learning stopped at different checkpoints yields little gain over using the base model itself. On the contrary, we see that the ensemble methods prove most beneficial when using a diverse set base learners. In practice, we recommand to use networks with different structures to enhance the diversity of the base learners.
- In all experiments, the SL performs better than unweighted averaging, but the improvement is small. However, such improvement is practically important: we see the improvement for the SL compared to the best learner is consistent in our experiments. In real machine learning applications where we usually need to process millions of images, such consistent improvement can make real impact. In addition, another benefit of the SL is that it can make the machine learning system more robust, as it automatically down weights the base learners with bad performance. It is easy to imagine if some of the base learners fail and make random prediction, it would highly influence the performance for unweighted averaging, while less influence the SL.

  In table 11, we consider the library consists two failed classifier, which only output random prediction, and the first two ResNet 110 in table 3. We can see the Super Learner has superior performance compared to all the other ensemble methods.
- Super Learner was initially proposed as a cross-validation based ensemble method. However, since training CNNs is computationnally intensive and that validation sets are typically large in image recognition tasks, we used the validation set of the neural networks for computing the weights of Super Learner (single-split cross-validation), instead of using conventional cross-validation (multiple-fold cross-validation). The structure is simple and could be easily extended. One potential extension of the linear-weighted Super Learner would be stacking several 1×1 convolutions with non-linear activation layers in between. This structure could mimic the cascading/hierarchical ensemble [39, 44]. Due to the small number of parameters, we expect this meta-learner would not overfit the validation set and thus would help improve the prediction. However this involves non-convex optimization and the results might not be stable. We leave this as future work.
- In this study, we focusd on the prediction accuracy for image classification, as it is easy to understand and is widely studied in the computer vision society. There are many other important tasks: for example, for medical imaging, people may be interested in estimating the proportion of abnormal tissues, instead of simply classifying images. In this setting, the task changed from classification to regression, and the ensemble methods may have different behavior. This is another important direction for future research.

References

[1]. Berger JO and Bock M, Combining independent normal mean estimation problems with unknown variances, The Annals of Statistics (1976), pp. 642–648.

[2]. Breiman L, Bagging predictors, Machine learning 24 (1996), pp. 123–140.

[3]. Breiman L, Stacked regressions, Machine learning 24 (1996), pp. 49–64.

[4]. Breiman L, Random forests, Machine learning 45 (2001), pp. 5–32.

[5]. Chambaz A, Zheng W, and van der Laan M, Data-adaptive inference of the optimal treatment rule and its mean reward. the masked bandit, U.C. Berkeley Division of Biostatistics Working Paper Series (2016).

[6]. Cho K, Van Merriënboer B, Gulcehre C, Bahdanau D, Bougares F, Schwenk H, and Bengio Y, Learning phrase representations using rnn encoder-decoder for statistical machine translation, arXiv preprint arXiv:1406–1078 (2014).

[7]. Choromanska A, Henaff M, Mathieu M, Arous GB, and LeCun Y, The Loss Surfaces of Multilayer Networks., in AISTATS. 2015.

[8]. Davies MM and van der Laan MJ, Optimal spatial prediction using ensemble machine learning, The international journal of biostatistics 12 (2016), pp. 179–201. [PubMed: 27130244]

[9]. Dietterich TG, Ensemble methods in machine learning, in International workshop on multiple classifier systems. Springer, 2000, pp. 1–15.

[10]. Efron B and Morris C, Combining possibly related estimation problems, Journal of the Royal Statistical Society. Series B (Methodological) (1973), pp. 379–421.

[11]. Freund Y and Schapire RE, A decision-theoretic generalization of on-line learning and an application to boosting, Journal of computer and system sciences 55 (1997), pp. 119–139.

[12]. Freund Y, Schapire RE, et al., Experiments with a new boosting algorithm, in ICML, Vol. 96 1996, pp. 148–156.

[13]. Friedman JH, Greedy function approximation: a gradient boosting machine, Annals of statistics (2001), pp. 1189–1232.

[14]. Goodfellow I, Bengio Y, and Courville A, Deep learning (2016).

[15]. Green EJ and Strawderman WE, A james-stein type estimator for combining unbiased and possibly biased estimators, Journal of the American Statistical Association 86 (1991), pp. 1001–1006.

[16]. Grover A and Leskovec J, node2vec: Scalable feature learning for networks, in Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining ACM, 2016, pp. 855–864.

[17]. He K, Zhang X, Ren S, and Sun J, Deep residual learning for image recognition, arXiv preprint arXiv:1512–03385 (2015).

[18]. He K, Zhang X, Ren S, and Sun J, Delving deep into rectifiers: Surpassing human-level performance on imagenet classification, in Proceedings of the IEEE International Conference on Computer Vision 2015, pp. 1026–1034.

[19]. Hothorn T, Bühlmann P, Dudoit S, Molinaro A, and van der Laan MJ, Survival ensembles, Biostatistics 7 (2006), pp. 355–373. [PubMed: 16344280]

[20]. Ioffe S and Szegedy C, Batch normalization: Accelerating deep network training by reducing internal covariate shift, arXiv preprint arXiv:1502–03167 (2015).

[21]. Ju C, Combs M, Lendle SD, Franklin JM, Wyss R, Schneeweiss S, and van der Laan MJ, Propensity score prediction for electronic healthcare dataset using super learner and high-dimensional propensity score method, U.C. Berkeley Division of Biostatistics Working Paper Series (2016), p. Working Paper 351.

[22]. Krizhevsky A and Hinton G, Learning multiple layers of features from tiny images, Technical report, University of Toronto (2009).

[23]. Krizhevsky A, Sutskever I, and Hinton GE, Imagenet classification with deep convolutional neural networks, in Advances in neural information processing systems. 2012, pp. 1097–1105.

[24]. Kuncheva LI, Whitaker CJ, Shipp CA, and Duin RP, Limits on the majority vote accuracy in classifier fusion, Pattern Analysis & Applications 6 (2003), pp. 22–31.

[25]. LeCun Y, Bengio Y, and Hinton G, Deep learning, Nature 521 (2015), pp. 436–444. [PubMed: 26017442]

[26]. Lin M, Chen Q, and Yan S, Network in network, arXiv preprint arXiv:1312–4400 (2013).

[27]. Luedtke AR and van der Laan MJ, Super-learning of an optimal dynamic treatment rule, The international journal of biostatistics 12 (2016), pp. 305–332. [PubMed: 27227726]

[28]. Luong MT, Pham H, and Manning CD, Effective approaches to attention-based neural machine translation, arXiv preprint arXiv:1508–04025 (2015).

[29]. Mitchell TM, Machine learning. 1997, Burr Ridge, IL: McGraw Hill 45 (1997), pp. 870–877.

[30]. Perozzi B, Al-Rfou R, and Skiena S, Deepwalk: Online learning of social representations, in Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining ACM, 2014, pp. 701–710.

[31]. Pirracchio R, Petersen ML, Carone M, Rigon MR, Chevret S, and van der Laan MJ, Mortality prediction in intensive care units with the super icu learner algorithm (sicula): a population-based study, The Lancet Respiratory Medicine 3 (2015), pp. 42–52. [PubMed: 25466337]

[32]. Polley EC and Van Der Laan MJ, Super learner in prediction, U.C. Berkeley Division of Biostatistics Working Paper Series (2010).

[33]. Rao J and Subrahmaniam K, Combining independent estimators and estimation in linear regression with unequal variances, Biometrics (1971), pp. 971–990.

[34]. Rubin DB and Weisberg S, The variance of a linear combination of independent estimators using estimated weights, Biometrika 62 (1975), pp. 708–709.

[35]. Russakovsky O, Deng J, Su H, Krause J, Satheesh S, Ma S, Huang Z, Karpathy A, Khosla A, Bernstein M, et al., Imagenet large scale visual recognition challenge, International Journal of Computer Vision 115 (2015), pp. 211–252.

[36]. Simonyan K and Zisserman A, Very deep convolutional networks for large-scale image recognition, arXiv preprint arXiv:1409–1556 (2014).

[37]. Sinisi SE, Polley EC, Petersen ML, Rhee SY, and van der Laan MJ, Super learning: an application to the prediction of hiv-1 drug resistance, Statistical applications in genetics and molecular biology 6 (2007).

[38]. Srivastava N, Hinton GE, Krizhevsky A, Sutskever I, and Salakhutdinov R, Dropout: a simple way to prevent neural networks from overfitting., Journal of Machine Learning Research 15 (2014), pp. 1929–1958.

[39]. Su Y, Shan S, Chen X, and Gao W, Hierarchical ensemble of global and local classifiers for face recognition, IEEE Transactions on Image Processing 18 (2009), pp. 1885–1896. [PubMed: 19556198]

[40]. Szegedy C, Liu W, Jia Y, Sermanet P, Reed S, Anguelov D, Erhan D, Vanhoucke V, and Rabinovich A, Going deeper with convolutions, in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition 2015, pp. 1–9.

[41]. Van Der Laan MJ and Dudoit S, Unified cross-validation methodology for selection among estimators and a general cross-validated adaptive epsilon-net estimator: Finite sample oracle inequalities and examples, U.C. Berkeley Division of Biostatistics Working Paper Series. (2003).

[42]. van der Laan MJ, Polley EC, and Hubbard AE, Super learner, Statistical applications in genetics and molecular biology 6 (2007).

[43]. Veit A, Wilber M, and Belongie S, Residual networks are exponential ensembles of relatively shallow networks, arXiv preprint arXiv:1605–06431 (2016).

[44]. Wang H, Cruz-Roa A, Basavanhally A, Gilmore H, Shih N, Feldman M, Tomaszewski J, Gonzalez F, and Madabhushi A, Cascaded ensemble of convolutional neural networks and handcrafted features for mitosis detection, in SPIE Medical Imaging. International Society for Optics and Photonics, 2014, pp. 90410B–90410B.

[45]. Wolpert DH, Stacked generalization, Neural networks 5 (1992), pp. 241–259.

[46]. Zeiler MD and Fergus R, Visualizing and understanding convolutional networks, in European Conference on Computer Vision. Springer, 2014, pp. 818–833.

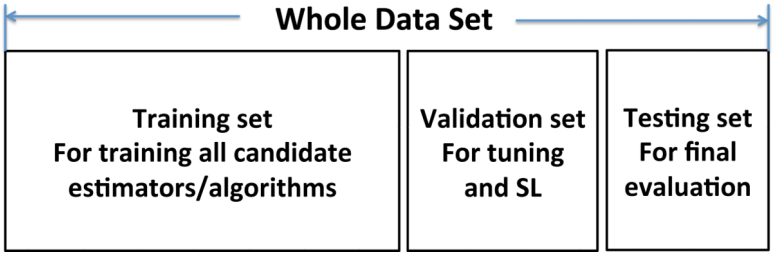

Figure 1.
Single Split (Sample Split) Super Learner, which computes the weights on the validation set.

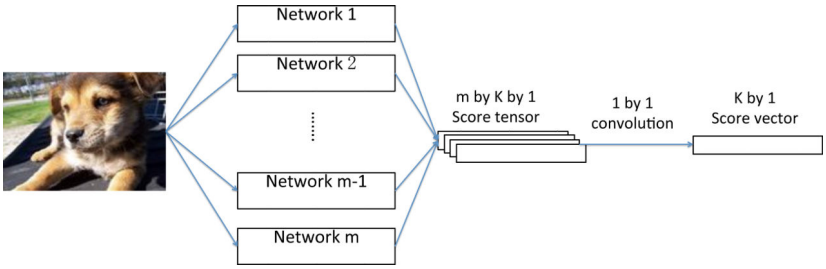

Figure 2.

Super Learner from convolution neural network perspective. The base learners are trained in the training set, and 1 by 1 convolutional layer is trained in the validation set. The simple structure of SL avoids the overfitting on the validation set.

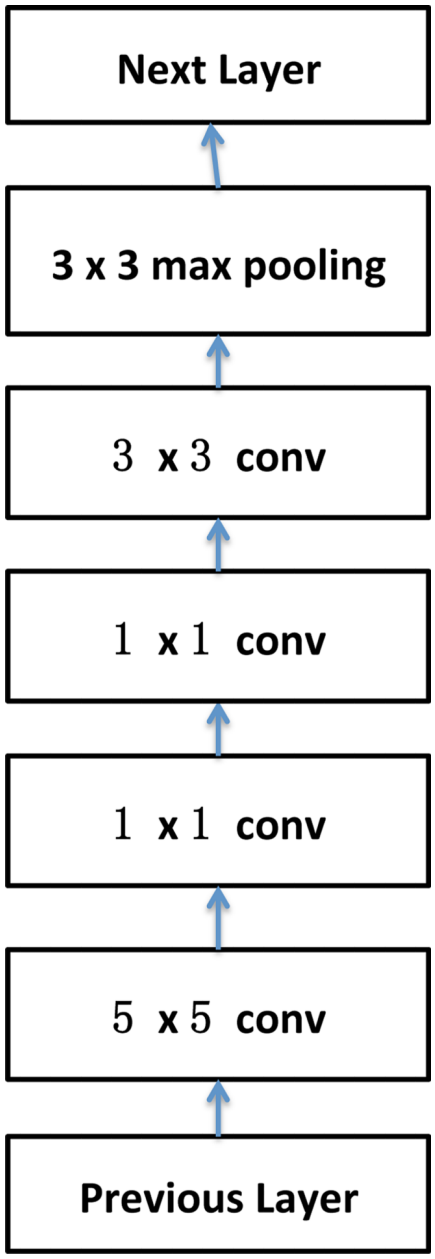

Figure 3.
An example of MLP layer in the NIN structure. Notice each convolution are followed by ReLU layer.

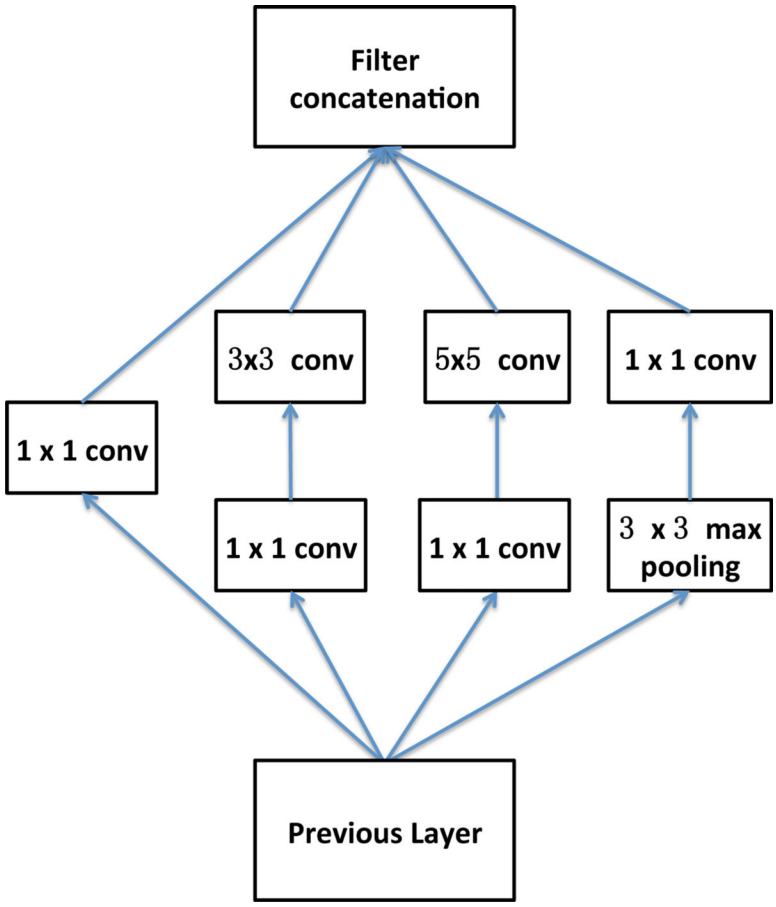

Figure 4.
An example of Inception module for GoogLeNet. Notice each convolution are followed by ReLU layer.

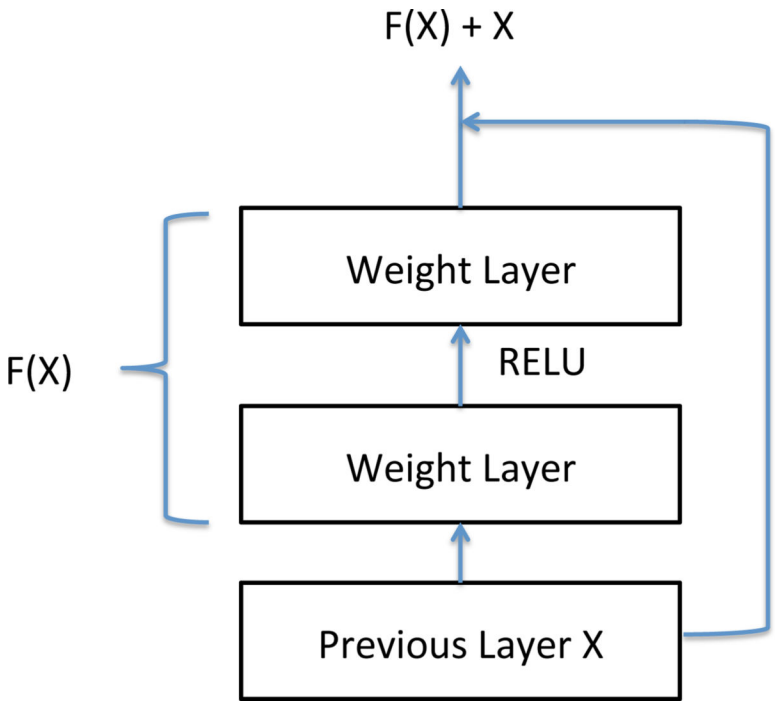

Figure 5.
An example of Inception module for GoogLeNet. Notice each convolution are followed by ReLU layer.

Table 1.

Left: Prediction accuracy on the testing set for ResNet 8 trained by 80, 90, 100, 110 epochs. Right: Prediction Accuracy on the testing set for ResNet 110 trained by 70, 85, 100, 115 epochs.

Training Epoch	Prediction Accuracy	Training Epoch	Prediction Accuracy
70	77.90%	70	88.96%
80	82.45%	85	89.99%
90	81.97%	100	93.18%
100	86.59%	115	93.54%

Table 2.

Prediction accuracy on the testing set for ResNet 8 and 110

Ensemble	ResNet 8	ResNet 110
Best Base Learner	86.59%	93.54%
SuperLearner	**86.79**%	**93.58**%
Discrete SuperLearner (nll)	86.59%	93.54%
Discrete SuperLearner (error)	86.59%	93.54%
Unweighted Average (before softmax)	86.11%	93.54%
Unweighted Average (after softmax)	86.14%	93.54%
BOC (before softmax)	86.59%	93.18%
BOC (after softmax)	86.59%	93.18%
Majority Voting	84.85%	93.19%

Table 3.

Prediction Accuracy on the testing set for ResNet with 8 and 110 layers

Model	Prediction Accuracy	Model	Prediction Accuracy
ResNet 8 0	87.85%	ResNet 110 0	93.99%
ResNet 8 1	88.19%	ResNet 110 1	93.64%
ResNet 8 2	87.58%	ResNet 110 2	93.49%
ResNet 8 3	87.61%	ResNet 110 3	93.95%

Table 4.

Prediction accuracy on the testing set for ensemble methods. The algorithm candidates are the ResNets with same structure but trained several times, where the differences come from randomized initialization and SGD.

Ensemble	ResNet 8	ResNet 110
Best Base Learner	88.20%	93.99%
SuperLearner	**90.73**%	**95.42**%
Discrete SuperLearner (nll)	88.20%	93.95%
Discrete SuperLearner (error)	87.61%	93.95%
BOC (before Sotmax)	88.20%	93.95%
BOC (after Sotmax)	88.20%	93.95%
Unweighted Average (before Sotmax)	90.68%	95.42%
Unweighted Average (afterbefore Sotmax)	90.68%	95.41%
Majority Vote	90.00%	95.10%

Table 5.

Prediction Accuracy on the testing set for networks with different structure

Model	Prediction Accuracy
NIN	86.77%
VGG	89.14%
ResNet 32	91.81%
ResNet 44	92.43%
ResNet 56	92.72%
ResNet 110	93.99%

Table 6.

Cross-entropy on the testing set for Networks with different structure

Model	Cross-entropy
NIN	0.5779
VGG	1.5649
ResNet 32	1.5442
ResNet 44	1.5341
ResNet 56	1.5327
ResNet 110	1.5242

Table 7.

Cross-entropy on the testing set for networks with different structure

Model	Image 1	Image 2	Image 3	Image 4	Image 5
NIN	0.9999	0.9999	0.09985	0.5306	1.000
VGG	0.2319	0.2319	0.2319	0.2302	0.2314
ResNet 32	0.2319	0.2318	0.2317	0.2316	0.2317

Table 8.

Prediction accuracy on the testing set for ensemble methods. The algorithm candidates include NIN, VGG, ResNet 32, ResNet 44, and ResNet 56. We compare the performance with/without the over-confident NIN network.

Ensemble	Without NIN	With NIN
Best Base Learner	93.99%	93.99%
SuperLearner	**94.69**%	**94.75**%
Discrete SuperLearner (nll)	93.99%	86.77%
Discrete SuperLearner (error)	93.99%	93.99%
BOC (before softmax)	93.99%	86.77%
BOC (after softmax)	93.99%	86.77%
Unweighted Average (before softmax)	94.56%	92.23%
Unweighted Average (after softmax)	94.55%	89.74%
Majority Vote	94.33%	94.13%

Table 9.

Prediction accuracy on the testing set for ensemble methods. The algorithm candidates include VGG, ResNet 32, ResNet 44, and ResNet 56. We compared the performance with/without five under-optimized GoogLeNets.

Ensemble	Without GoogLeNet	With 3 GoogLeNets	With 5 GoogLeNets
Best Base Learner	93.99%	93.99%	93.99%
SuperLearner	**94.75**%	**94.77**%	**94.77**%
Discrete SuperLearner (nll)	93.99%	93.99%	93.99%
Discrete SuperLearner (error)	93.99%	93.99%	93.99%
BOC (before softmax)	93.99%	93.99%	93.99%
BOC (after softmax)	93.99%	93.99%	93.99%
Unweighted Average (before softmax)	94.56%	93.26%	90.01%
Unweighted Average (after softmax)	94.55%	93.29%	90.07%
Majority Vote	94.33%	92.63%	87.20%

Table 10.

Prediction accuracy on the testing set for all the ensemble methods using all the networks mentioned in this study as base learners.

Ensemble	Accuracy
Best base learner	93.99%
SuperLearner	**95.02**%
Discrete SuperLearner (nll)	93.95%
Discrete SuperLearner (error)	93.95%
BOC (before softmax)	93.95%
BOC (after softmax)	93.95%
Unweighted Average (before softmax)	94.44%
Unweighted Average (after softmax)	94.48%
Majority Vote	94.10%

Table 11.

Prediction accuracy on the testing set for ensemble methods with two ResNet 110 and 2 random classifier.

Ensemble	Testing Accuracy
Best Base Learner	93.99%
SuperLearner	**94.90**%
Discrete SuperLearner (nll)	93.99%
Discrete SuperLearner (error)	93.99%
BOC (before Sotmax)	93.99%
BOC (after Sotmax)	93.99%
Unweighted Average (before Sotmax)	86.54%
Unweighted Average (afterbefore Sotmax)	86.33%
Majority Vote	89.16%