J Intell Inf Syst
https://doi.org/10.1007/s10844-018-0516-5

Experimental validation for N-ary error correcting output codes for ensemble learning of deep neural networks

Kaikai Zhao[1] · **Tetsu Matsukawa**[2] · **Einoshin Suzuki**[2]

Received: 13 September 2017 / Revised: 26 June 2018 / Accepted: 29 June 2018

Abstract N-ary error correcting output codes (ECOC) decompose a multi-class problem into simpler multi-class problems by splitting the classes into N subsets (meta-classes) to form an ensemble of N-class classifiers and combine them to make predictions. It is one of the most accurate ensemble learning methods for traditional classification tasks. Deep learning has gained increasing attention in recent years due to its successes on various tasks such as image classification and speech recognition. However, little is known about N-ary ECOC with deep neural networks (DNNs) as base learners, probably due to the long computation time. In this paper, we show by experiments that N-ary ECOC with DNNs as base learners generally exhibits superior performance compared with several state-of-the-art ensemble learning methods. Moreover, our work contributes to a more efficient setting of the two crucial hyperparameters of N-ary ECOC: the value of N and the number of base learners to train. We also explore valuable strategies for further improving the accuracy of N-ary ECOC.

Keywords N-ary error correcting output codes · Ensemble learning · Deep neural networks · Image classification

✉ Kaikai Zhao
cyoukaikai@gmail.com

Tetsu Matsukawa
matsukawa@inf.kyushu-u.ac.jp

Einoshin Suzuki
suzuki@inf.kyushu-u.ac.jp

[1] Graduate School of Systems Life Sciences, Kyushu University, Fukuoka, Japan

[2] Department of Informatics, ISEE, Kyushu University, Fukuoka, Japan

Published online: 07 July 2018

1 Introduction

Classification is a data mining task of predicting predefined classes. It is generally conducted by constructing a model (classifier) based on training data and using it to classify new test data. Image classification refers to classifying images according to their visual contents to classes of interest. There are many important image classification applications such as medical diagnosis (Müller et al. 2004), object recognition (Russakovsky et al. 2015), face recognition (Ahonen et al. 2006) and person re-identification (Ahmed et al. 2015).

Hand-designed features were usually used to classify images before the success of deep neural networks (DNNs) at the ImageNet Large Scale Visual Recognition Challenge (ILSVRC, Russakovsky et al. (2015)). In the ILSVRC2012 competition, a DNN called AlexNet (Krizhevsky et al. 2012) outperformed the other methods, which typically used hand-designed features, by more than 10%. Since then, the ILSVRC competitions are consistently won by DNNs (Goodfellow et al. 2016).

Ensemble learning constructs a number of base learners and combines them to make predictions. It is a powerful technique to improve the classification accuracy. For instance, Krizhevsky et al. (2012) reported that the error rate of a single AlexNet was 18.2%, while with an ensemble of seven AlexNet models, the error rate dropped to 15.3%. The reason ensemble learning generally exhibits a high accuracy is that different base learners usually do not make the same errors on the test set (Goodfellow et al. 2016).

There are two main approaches for designing an ensemble learning algorithm (Dietterich 2002). The first approach aims to construct independent models that are accurate and diverse. The second approach learns different models in a "coupled fashion" such that the interactions among different models are emphasized.

For the first approach, independent models are usually trained using the following strategies: a) randomly sampling the training examples (e.g., Bagging, Breiman (1996)) and b) randomly selecting the input features (e.g., random forest (Ho 1995)). Besides these strategies, independent models can also be obtained by manipulating the class labels of the training data, for instance, error correcting output code (ECOC, Dietterich and Bakiri (1995)) and N-ary ECOC (Zhou et al. 2016).

ECOC was first designed to correct errors caused by noise in the channel during data transmission in communication tasks (Han et al. 2011). The idea of the error correction is to add redundancy to the data when transmitting them so that even if some errors occur we are still able to correct them. For classification tasks, ECOC forms an ensemble of a number of binary classifiers by randomly splitting the classes of a multi-class problem into two subsets (meta-classes, each of which is a super-class composed of classes).

N-ary ECOC is an extension of ECOC to the case of N meta-classes, as binary encoding (the way of splitting the classes) may result in difficult classification problems if the between-class difference is subtle (Zhou et al. 2016). For instance, in the fine-grained classification, the classes are very similar and difficult to discriminate from each other (Zhou et al. 2016).

For the second approach, different models are usually trained by somehow interacting with one another. Boosting (Schapire 1990) trains a number of models sequentially with the later models compensating the mistakes made by the earlier models. One popular Boosting method is AdaBoost (Freund and Schapire 1997), which increases the weights of the misclassified examples in the earlier models so that the later models focus on the difficult examples in the training set.

There are several ensemble learning methods using DNNs as base learners. For instance, one often used method is to train a number of DNNs with different random initializations (Wei et al. 2015), of which idea is to inject randomness into the models. We refer to this method as RI, which stands for random initialization. Horizontal voting (Xie et al. 2013) trains a single DNN until convergence and then continues the training with additional iterations, meanwhile it saves the snapshots along the path to get a pool of base learners.

The previous work of N-ary ECOC (Zhou et al. 2016) exhibited superior performance on several datasets using support vector machines (SVMs, Cortes and Vapnik (1995)) or decision trees (Quinlan 1986) as base learners, compared with ECOC using several state-of-the-art encoding strategies. However, N-ary ECOC with DNNs as base learners remains unexplored, probably because training DNNs is generally much more computationally expensive than training traditional classifiers. Modern DNNs can take hours or days to train. Thus heavily tuning the two hyperparameters of N-ary ECOC: N (the number of meta-classes) and N_L (the number of base learners), has not been an attractive idea for DNN base learners.

In this paper, we investigate three important issues regarding N-ary ECOC with DNNs as base learners. We focus on deep learning of image classification tasks, as deep learning is very successful on image classification, especially when the number of classes is large. 1) Which ensemble learning methods to use? This issue is important because using DNNs as base learners for N-ary ECOC remains unexplored. Moreover, the previous work of N-ary ECOC (Zhou et al. 2016) was compared with only ECOC using several state-of-the-art encoding strategies. It is not clear if N-ary ECOC can compete with RI, Bagging, AdaBoost and other state-of-the-art ensemble learning methods. 2) Assuming N-ary ECOC is the best method, how to choose the two crucial hyperparameters (N and N_L) for N-ary ECOC? Better values for N might exist outside the region recommended by Zhou et al. (2016) when using DNNs as base learners, and the range of the optimal value for N_L was not given by Zhou et al. (2016). 3) Can we use RI or HV to improve the ensemble accuracy of N-ary ECOC? RI or HV can be used to enhance a single base learner of N-ary ECOC and thus has the potential of improving the overall ensemble accuracy of N-ary ECOC.

2 Related work

ECOC forms an ensemble of binary classifiers by randomly splitting the classes into two meta-classes and combining the classifiers to make predictions. It consists of two main steps: encoding and decoding. In the encoding step, a codeword matrix $M \in \{0,\ 1\}^{N_C \times N_L}$ is first generated, where N_C is the number of classes of the original multi-class problem and N_L is the codeword length, which corresponds to the number of base learners to train. Each class is assigned with a unique bit vector called a "codeword" of length N_L. Table 1 (a) shows an example of 8-bit codewords assigned to classes c_0, ..., c_9. Each row of M is the codeword of a class. For instance, the codeword of c_0 is [0 1 1 1 0 1 0 0]. Each column of M represents a bit position of the codewords and specifies how to split the classes into two meta-classes. For instance, the meta-class 0 of f_0 consists of c_0, c_2, c_5, c_7 and c_9, and the meta-class 1 of f_0 consists of c_1, c_3, c_4, c_6 and c_8. Then a binary classifier is learned to predict the two meta-classes.

In the decoding step, ECOC compares the prediction vector, which consists of the predicted meta-class labels for each of the binary classifiers, with the codeword of each class

Table 1 Examples of codeword matrix for ECOC, N-ary ECOC and RI

Class	bit positions of the codewords							
	f_0	f_1	f_2	f_3	f_4	f_5	f_6	f_7
(a) ECOC encoding								
c_0	0	1	1	1	0	1	0	0
c_1	1	1	0	1	0	1	1	0
c_2	0	0	1	0	0	0	1	0
c_3	1	1	0	1	1	0	0	1
c_4	1	0	1	0	1	0	1	0
c_5	0	0	0	0	1	1	0	1
c_6	1	1	1	0	0	1	1	0
c_7	0	1	1	1	1	0	0	1
c_8	1	0	0	0	0	0	1	1
c_9	0	0	0	1	1	1	0	1
(b) N-ary ECOC encoding ($N = 5$)								
c_0	0	1	4	0	1	4	4	2
c_1	2	3	3	1	4	1	0	1
c_2	2	2	0	3	2	2	0	1
c_3	3	2	1	4	0	3	2	3
c_4	1	3	2	1	4	3	1	3
c_5	1	4	2	4	3	4	2	2
c_6	3	4	0	2	1	1	4	0
c_7	4	0	1	0	3	0	3	0
c_8	0	0	4	3	2	2	1	4
c_9	4	1	3	2	0	0	3	4
(c) RI encoding								
c_0	0	0	0	0	0	0	0	0
c_1	1	1	1	1	1	1	1	1
c_2	2	2	2	2	2	2	2	2
c_3	3	3	3	3	3	3	3	3
c_4	4	4	4	4	4	4	4	4
c_5	5	5	5	5	5	5	5	5
c_6	6	6	6	6	6	6	6	6
c_7	7	7	7	7	7	7	7	7
c_8	8	8	8	8	8	8	8	8
c_9	9	9	9	9	9	9	9	9

to decide the "closest" one as the final prediction for a test example. The commonly used distance measure is Hamming distance, which counts the number of different elements between two vectors. Take the codeword matrix of Table 1a for example, if the prediction vector for a test example (whose real class label is c_0) is [0 1 1 0 0 1 0 0], then the closest codeword to this prediction vector is [0 1 1 1 0 1 0 0] (the codeword of c_0), where only the

fourth bit position is different from the prediction vector. Even if a binary classifier makes an error on a test example, the error still has a good chance to be corrected by other binary classifiers because of the redundancy of additional bits in the codeword matrix. In the above example, the fourth binary classifier makes a mistake, i.e., it predicts meta-class 0 while the real meta-class label (the fourth element of the codeword of c_0) is 1. However, as all the other seven classifiers make correct predictions, the error is finally corrected.

N-ary ECOC (Zhou et al. 2016) extends ECOC to the case of N meta-classes. It solves a multi-class problem by decomposing it into a number of N-class ($3 \leq N < N_C$) problems and combining them to make predictions. It has one main advantage compared with ECOC: it enables designing more discriminative codes.

N-ary ECOC also consists of two main steps: encoding and decoding. In the encoding step, a codeword matrix $M \in \{0, \ ..., \ N-1\}^{N_C \times N_L}$ is first generated. Table 1b shows an example of 8-digit N-ary ECOC ($N = 5$) codeword matrix. Each row of M is the codeword of a class. Each column of M represents how to split the classes into N meta-classes, then an N-class classifier is learned to predict one of the N meta-classes. In the decoding step, N-ary ECOC compares the prediction vector with each of the codewords to decide the closest one as the final prediction. For instance, if the prediction vector for N-ary ECOC using the codeword matrix in Table 1b is [0 3 4 0 1 3 4 2] for a test example, then c_0 (with the codeword [0 1 4 0 1 4 4 2]) is selected as the final prediction, as it has the minimum Hamming distance to the prediction vector.

Note that RI, the ensemble learning method that trains a number of DNNs with different random initializations, can also be viewed as training its base learners based on a codeword matrix but with identical codes (as shown in Table 1c).

There are two crucial hyperparameters involved in N-ary ECOC: the value of N and the codeword length N_L. The previous work of N-ary ECOC (Zhou et al. 2016) showed that a large value for N leads to a better quality of codeword matrix in terms of *column correlations* and *row separations*. Here the column correlations are measured by Pearson's correlation (Pearson 1895), which is the covariance of two variables (the elements of a column are regarded as the samples of a variable) divided by the product of their standard deviations. The row separations are measured by the minimum Hamming distance between the rows of the codeword matrix.

However, Zhou et al. (2016) further argued that a large value for N also increases the classification difficulty (measured by the test error) of the base learners of N-ary ECOC as they need to discriminate more meta-classes. If there are sufficient base learners, the disadvantage (the classification difficulty) outweighs the benefit (the good quality of the codeword matrix) gained from a large value for N. Finally, they recommended a value for N in the range of 3 to 10 for N-ary ECOC to achieve its maximum ensemble accuracy with respect to all possible values for N.

However, the increasing classification difficulty of the base learners of N-ary ECOC caused by a relatively large value for N does not necessarily result in a low classification accuracy for DNNs. DNNs automatically learn representative features from data, while traditional classifiers use hand-designed features as input (Goodfellow et al. 2016). The base learners of N-ary ECOC with a small value for N will no longer need to discriminate a large number of classes under the same meta-class. As a simple example, the DNN base learners of N-ary ECOC with $N = 3$ for a 1,000-class problem (e.g., ILSVRC2012 (Russakovsky et al. 2015) dataset) will no longer need to learn features to discriminate approximately 333 classes under the same meta-class. How this task setting affects the learning of representative features for DNNs, and in turn, affects the classification accuracy of the trained base learners are unknown.

Table 2 Statistics of the tested datasets and DNN models

Dataset	Image Size	No. Train	No. Test	N_C	Model	Tested N	Tested N_L
MNIST	28×28	60,000	10,000	10	MNIST-LeNet	3,4,5,7	60
SVHN	32×32	73,257	26,032	10	SVHN	3,4,5,7	100
CIFAR10	32×32	50,000	10,000	10	CIFAR10-QUICK	3,4,5,7	100
					CIFAR10-FULL	3,4,5,7	100
CIFAR100	32×32	50,000	10,000	100	CIFAR100-NIN	3,5,25,33,50,75,95	60
					CIFAR100-QUICK	3,4,5,10,25,33,50,75,95	100
FLOWER102	256×256	1,020	1,020	102	FLOWER102-AlexNet	3,5,10,34,51,95	60

Here N_C is the number of classes of the original multi-class problem. N is the number of meta-classes for the base learners of N-ary ECOC. N_L is the number of base learners to train in the ensemble learning and also means the codeword length for ECOC and N-ary ECOC

As for the optimal value for N_L for a specific N, the previous work of N-ary ECOC (Zhou et al. 2016) did not suggest a clear answer. In their experiments, they compared N-ary ECOC with ECOC using a codeword length of $N_L \leq N_C(N_C-1)/2$. They argued that it is sufficient for ECOC to reach its optimal performance (Allwein et al. 2000), and N-ary ECOC needs a relatively small number of base learners than ECOC does as N-ary ECOC has more discriminative codes (Zhou et al. 2016). However, $N_C(N_C-1)/2$ is a very large value when N_C is large. For instance, given a 100-class classification problem, we need to train approximately 4,950 base learners. Such a situation is very demanding even for traditional classifiers. DNNs are generally much more computationally expensive to train. Thus, we need to find a smaller value for N_L for N-ary ECOC with DNNs as base learners.

3 Experimental design

We conduct all the experiments using the Caffe (Jia et al. 2014) framework.

3.1 Datasets and DNN models

We use five datasets, MNIST (LeCun et al. 1998), SVHN (Netzer et al. 2011), CIFAR10 (Krizhevsky and Hinton 2009), CIFAR100 (Krizhevsky and Hinton 2009) and FLOWER102 (Nilsback and Zisserman 2008). Table 2 shows the statistics of the datasets.

We employ seven public off-the-shelf DNN models (as listed in Table 2). Among them, MNIST-LeNet, CIFAR10-QUICK, CIFAR10-FULL and CIFAR100-QUICK are built-in models in the Caffe (Jia et al. 2014) framework, while the other models (SVHN,[1] CIFAR100-NIN[2] and FLOWER102-AlexNet[3]) are public models available online. Note that FLOWER102-AlexNet is obtained by fine-tuning a pre-trained AlexNet (using the ILSVRC2012 dataset) on the FLOWER102 dataset. This fine-tuning is necessary because

[1] https://github.com/alexvking/neural-net-house-number-recognition

[2] https://gist.github.com/mavenlin/e56253735ef32c3c296d

[3] https://github.com/jimgoo/caffe-oxford102

the training data size of FLOWER102 is too small and thus it is difficult to directly learn representative features from the training data, while the pre-trained AlexNet can provide well-learned features.

We follow exactly the same settings for the employed MNIST-LeNet, CIFAR10-FULL, SVHN and FLOWER102-AlexNet models, and slightly modify the training procedures for the remaining models. The details and the reasons for the modification are described below.

We set the batch size of CIFAR10-QUICK to 350 to make it exactly the same as Lee et al. (2015) for comparison purpose. The CIFAR100-QUICK model is modified from the CIFAR10-QUICK model. We set the number of output units of its last layer to 100 to fit the number of classes of the CIFAR100 dataset. We observed that the training loss continues to decrease by a large amount for CIFAR100-QUICK if it is trained for 5,000 iterations as in CIFAR10-QUICK. This decrease is probably because CIFAR100 has more classes. Thus we adopt a relatively large number of iterations (compared with CIFAR10-QUICK) to ensure that CIFAR100-QUICK converges, i.e., the training loss stops further dropping. The modified settings are: we use a learning rate of 0.001 in the first 8,000 iterations and 0.0001 in the next 2,000 iterations. Note that the learning rate of CIFAR10-QUICK is 0.001 in the first 4,000 iterations and 0.0001 in the next 1,000 iterations.

The CIFAR100-NIN model was first introduced by Lin et al. (2014). We slightly modified the settings of its parameters for fast convergence. Note that it takes approximately 5.5 hours to train a single model with an NVIDIA TITAN X GPU (modern DNNs use GPU to conduct most of the computing (Goodfellow et al. 2016)) if we follow the original training procedure as described in Lee et al. (2015). If we train 60 models for RI, Bagging, AdaBoost and ECOC, and investigate seven different values for N for N-ary ECOC (the experiments will be given in Sections 4.1 and 4.2), it takes approximately 151 days to finish the experiments. Thus we use an initial learning rate of 0.01 and decay the learning rate by a factor of 10 for every 20,000 iterations. The network is trained for 50,000 iterations with a batch size of 100. Under the above settings, the base models exhibit an average accuracy of 59.79% (0.40% lower than the reported 60.19% by Lee et al. (2015)). However, we can train approximately eight models with this modified training procedure instead of a single model with the original training procedure (Lee et al. 2015).

3.2 Settings for ECOC and N-ary ECOC

ECOC and N-ary ECOC train N_L base learners based on their codeword matrices (encoding) and make predictions based on a distance measure (decoding). Basically, there are many encoding and decoding methods designed for general and task-specific applications (see Zhou et al. (2016) for a review). However, as the focus of this paper is to investigate the three issues as mentioned at the end of Section 1, we simply choose the most frequently used encoding and decoding methods. We employ random dense encoding (Allwein et al. 2000), which randomly splits the classes into two subsets for ECOC. The idea is applied to N-ary ECOC, which randomly splits the classes into N subsets.

One notable thing for designing the codeword matrix is that, when we use DNNs as base learners, the number of classes in each meta-class should be approximately balanced. Masko and Hensman (2015) reported that a skewed class distribution can significantly decrease the performance of deep learning. Thus, we explicitly enforce a balanced encoding of the classes. That is, each meta-class consists of either $\lfloor N_C/N \rfloor$ classes or $\lceil N_C/N \rceil$ classes. Here $\lfloor x \rfloor$ is the *floor* function of x, which represents the greatest integer less than or equal to x, and $\lceil x \rceil$ is the *ceiling* function of x, which represents the smallest integer greater than or equal to x.

The minimum Hamming distance is very popular and simple as the decoding method for both ECOC and N-ary ECOC and thus we adopt it. The tested values for N_L in ECOC and N-ary ECOC, as well as the tested values for N in N-ary ECOC, are listed in Table 2.

4 Experimental results

4.1 Issue 1: What ensemble learning methods to use?

Compared methods We compare the following methods: SM (training only a single model for the original N_C-class classification problem), RI, Bagging, AdaBoost, ECOC, N-ary ECOC and two other state-of-the-art methods (i.e., stochastic multiple choice learning (sMCL, Lee et al. (2015)) and DivLoss (Opitz et al. 2016)). Here SM is served as a baseline to know how much each ensemble learning method improves the accuracy over training only a single model.

sMCL (Lee et al. 2015) learns all its base learners simultaneously. During the training, it assigns an example to the base learner that exhibits the smallest loss and employs a "winner-take-gradient" strategy to update the gradient. As a result, different base learners are specialized and diversified with a subset of training examples. During the testing, the prediction of the base learner with the smallest loss is chosen as the final prediction for a test example. Opitz et al. (2016) proposed to divide the neurons of the last layer of a DNN into several groups. Each group together with all the previous layers forms a classifier. They proposed "DivLoss" to explicitly enforce each individual group to make diverse predictions. The DivLoss consists of two terms: the cross entropy of the prediction errors of the individual base learners and the cross entropy of the predictions between all base learner pairs.

For the AdaBoost method, several variants exist (Hastie et al. 2009) and most of them vary in how to increase the weights of the misclassified examples. We employ AdaBoost.SAMMAR (Hastie et al. 2009) because it is very effective in dealing with multi-class problems. The base learners of AdaBoost in general can be trained by either *reweighting* or *resampling* (Freund and Schapire 1997). Reweighting takes the weights of the training examples into account when estimating the loss. Resampling generates a new training set by sampling the original training set based on the weights, which are regarded as a probability distribution, a base learner is then trained on the sampled dataset. It is argued that there is no significant difference between reweighting and resampling in terms of ensemble accuracy (Zhou 2012). However, Seiffert et al. (2008) showed through extensive experiments that resampling is preferred over reweighting. Thus we choose resampling.

Note that we do not include random forest in the comparison, because it typically requires features as input while we tackle image classification with DNNs, which learn features from raw data.

Class balance and class imbalance We evaluate the compared methods on two typical classification tasks: *class balanced* and *class imbalanced*. In the class balanced task, the number of examples in each class is roughly the same. In the class imbalanced task, the number of examples in some classes (*minority classes*) is significantly smaller than those of other classes (*majority classes*). The issue of class imbalance frequently occurs in real-world applications. Take medical diagnosis (Müller et al. 2004) as an example, the patients who are diagnosed as having a specific disease (positive class) are usually very few in number, compared with those who are diagnosed as no disease (negative class).

Buda et al. (2017) suggested that there are two types of class imbalance which are representative in real-world applications. The first one is that the number of examples is equal in minority classes and equal in majority classes but differs between the minority classes and the majority classes. They refer to this type as *step imbalanced*. The second one is that the sorted numbers of examples for the classes roughly linearly increase. For instance, let $|c_i|$, where $i \in \{0, \ ..., \ N_C - 1\}$, be the number of examples belonging to class c_i, then $(|c_i|)$ could be (10, 20, 30, ...). They refer to this type as *linear imbalanced*.

The datasets listed in Table 2 belong to the class balanced tasks. To generate the step imbalanced and the linear imbalanced datasets, we down-sample the original CIFAR10 and CIFAR100 datasets, which have a small value for N_C and a large value for N_C, respectively.

Following the definitions of Buda et al. (2017), let μ be the fraction of minority classes, and $\mu = N_{minority}/N_C$, where $N_{minority}$ is the number of minority classes; let ρ be the imbalance ratio, and $\rho = min(|c_i|)/max(|c_i|)$. For the step imbalanced data, we first randomly select 10% and 50% of the N_C classes as the minority classes. Then for each of the minority classes, we randomly select 10% and 20% of its examples from both the training data and test data to keep their minority classes identical. Note that the above settings refer to *task 1*: ($\mu = 0.1, \ \rho = 0.1$), *task 2*: ($\mu = 0.1, \ \rho = 0.2$), *task 3*: ($\mu = 0.5, \ \rho = 0.1$) and *task 4*: ($\mu = 0.5, \ \rho = 0.2$). For the linear imbalanced data, we first shuffle the N_C classes, then randomly sample $j \times max(|c_i|)/N_C$ examples for each of the shuffled classes, where j is the index of the classes in the shuffled order. We call this task *task 5*.

Evaluation on class balanced tasks In order to provide a robust evaluation, we conduct K-fold cross validation. Because the typical training-test splits listed in Table 2 are widely used in the deep learning community, we set $K = \lceil (\text{No. Train} + \text{No. Test})/\text{No. Test} \rceil$ for each dataset, where No. Train and No. Test are the numbers of examples in the training data and test data, respectively. For instance, the MNIST dataset has 60,000 training examples and 10,000 test examples, thus $K = 7$.

We report both ensemble accuracy (EA) and oracle accuracy (Vriesmann et al. 2012) for the compared ensemble learning methods. The EA, which is the accuracy of the combined votes, is often estimated based on either the predicted class labels or the posterior probabilities. Here we estimate the EA based on the predicted class labels due to its simplicity. Oracle accuracy is the ratio of the examples in the test data correctly classified by at least one classifier of an ensemble. We use oracle accuracy because it is an upper limit of the ensemble accuracy. An ensemble learning method with a very low oracle accuracy is unlikely to be significant, because it indicates that all the base learners make wrong predictions on a large portion of the test examples.

Table 3 lists the EAs of different methods. We see several interesting facts. First, N-ary ECOC outperforms all other methods in most cases, especially when the number of classes is small (i.e., $N_C < 25$). The possible reasons for the superior ensemble accuracy and the effective range of N-ary ECOC will be given below and in Section 4.2, respectively.

Second, N-ary ECOC always outperforms ECOC. Zhou et al. (2016) argued that it is mainly due to the better quality of the codeword matrix and the higher discriminative ability (in terms of how many meta-classes a base learner tries to discriminate) of the codes of N-ary ECOC compared with ECOC. However, we argue that the small degree of merging classes (measured by $N_C - N$, i.e., the number of classes decreased when the classes are merged into meta-classes) contributes more to the high ensemble accuracy of N-ary ECOC. More details will be given in Section 4.2.

Third, N-ary ECOC outperforms RI in all the tested DNNs except CIFAR100-NIN and CIFAR100-QUICK. We argue that it is mainly because N-ary ECOC generally makes more

Table 3 Ensemble accuracies with their standard deviations in the K-fold cross validation

Model	SM	Ensemble Accuracy					
		RI	Bagging	AdaBoost	ECOC	N-ary ECOC	Other
MNIST-LeNet$_{\times 60}$	98.93$_{\pm 0.12}$	99.03$_{\pm 0.12}$	99.03$_{\pm 0.12}$	98.98$_{\pm 0.11}$	99.12$_{\pm 0.08}$	**99.16**$_{\pm 0.09}$	
SVHN$_{\times 100}$	90.74$_{\pm 0.12}$	92.23$_{\pm 0.12}$	92.24$_{\pm 0.16}$	91.56$_{\pm 0.11}$	92.63$_{\pm 0.32}$	**92.74**$_{\pm 0.30}$	
CIFAR10-QUICK$_{\times 100}$	76.31$_{\pm 0.72}$	82.04$_{\pm 0.44}$	81.94$_{\pm 0.31}$	77.44$_{\pm 0.57}$	81.52$_{\pm 0.59}$	**82.20**$_{\pm 0.44}$	
CIFAR10-FULL$_{\times 100}$	81.39$_{\pm 0.65}$	84.29$_{\pm 0.51}$	84.01$_{\pm 0.40}$	70.64$_{\pm 6.71}$	84.23$_{\pm 0.59}$	**84.51**$_{\pm 0.54}$	
CIFAR100-NIN$_{\times 60}$	59.36$_{\pm 0.30}$	**67.10**$_{\pm 0.19}$	65.36$_{\pm 0.69}$	55.31$_{\pm 0.35}$	38.97$_{\pm 0.32}$	66.81$_{\pm 0.18}$	66.51 (Lee et al. 2015)
CIFAR100-QUICK$_{\times 100}$	44.38$_{\pm 0.29}$	**56.37**$_{\pm 0.49}$	55.35$_{\pm 0.45}$	47.81$_{\pm 0.30}$	43.59$_{\pm 0.38}$	56.28$_{\pm 0.52}$	50.32 (Opitz et al. 2016)
FLOWER102-AlexNet$_{\times 60}$	80.89$_{\pm 0.31}$	83.66$_{\pm 0.62}$	83.22$_{\pm 0.67}$	83.66$_{\pm 0.62}$	66.97$_{\pm 0.90}$	**84.20**$_{\pm 0.28}$	

Here N = 3, 3, 7, 4, 95, 95, 95 for the tested DNN models from the top row to the bottom row

diverse predictions than RI. Ensemble learning benefits more if the base learners make very different (or diverse) errors compared with similar errors on the test set. Thus diverse predictions are desirable for achieving a relatively high ensemble accuracy. The diversity is measured by the degree of difference in the predictions of base learners on the test set. Here we use the Q statistic (Yule 1900), which is one of the most frequently used diversity measures, to evaluate the diversity of the trained base learners. The Q statistic of the pairwise diversity between two classifiers C_i and C_j is given by

$$Q_{i,j} = \frac{N^{11}N^{00} - N^{01}N^{10}}{N^{11}N^{00} + N^{01}N^{10}}, \tag{1}$$

where N^{11} and N^{00} are the numbers of examples correctly and incorrectly predicted by both C_i and C_j, respectively; N^{01} and N^{10} are the numbers of examples correctly predicted by only C_i and only C_j, respectively. The Q value, which is between [-1, 1], is smaller if the two classifiers are more diverse. The average Q statistic Q_{avg} over all pairs of classifiers is given by

$$Q_{avg} = \frac{2}{N_L(N_L-1)} \sum_{i=1}^{N_L-1} \sum_{j=i+1}^{N_L} Q_{i,j}, \tag{2}$$

where N_L is the total number of classifiers. Table 4 lists Q_{avg} of the tested DNNs for both RI and N-ary ECOC. We see that N-ary ECOC always exhibits smaller values for Q_{avg} compared with RI.

Fourth, the lower the single model accuracy, the more the DNN benefits from ensemble learning. For instance, RI improves the accuracy over SM by 0.10% and 11.99% for MNIST-LeNet and CIFAR100-QUICK, respectively. The large improvement of the accuracy for DNNs with low SM accuracies is probably because a relatively large portion of incorrect predictions are corrected by ensemble learning, compared with DNNs with high SM accuracies.

Fifth, RI generally outperforms Bagging and AdaBoost. Lee et al. (2015) argued that "the loss of unique data is the primary negative effect of bagging". Bagging samples the training data with replacement. Therefore, only approximately 63.2% unique data are reserved in the sampled dataset (Han et al. 2011). As a result, the trained base models of Bagging have lower accuracies compared with those of RI. In our experiments, we observed that the average unique examples resampled by the base learners of AdaBoost are only 13.7% of the original training dataset for CIFAR10-QUICK. The possible reason is that AdaBoost keeps increasing the weights of the difficult examples, which are usually a small portion of the

Table 4 Average Q statistics (Q_{avg}) of RI and N-ary ECOC

Model	Q_{avg}	
	RI	N-ary ECOC
MNIST-LeNet$_{\times 60}$	0.998	**0.976**
SVHN$_{\times 100}$	0.977	**0.863**
CIFAR10-QUICK$_{\times 100}$	0.876	**0.802**
CIFAR10-FULL$_{\times 100}$	0.941	**0.779**
CIFAR100-NIN$_{\times 60}$	0.881	**0.876**
CIFAR100-QUICK$_{\times 100}$	0.822	**0.813**
FLOWER102-AlexNet$_{\times 60}$	0.963	**0.958**

Here N = 3, 3, 7, 4, 95, 95, 95 for the tested DNN models. Note that the smaller the Q_{avg}, the more diverse the predictions

whole training dataset. As a result, the easy examples become less and less likely to have a chance to contribute to the learning of a DNN. Thus the issue of the loss of unique data for AdaBoost may be more serious than Bagging.

Table 5 lists the oracle accuracy of the compared methods. We see that ECOC and N-ary ECOC achieve 100% or close to 100% oracle accuracy, and generally outperform all other methods. This fact is probably because once a meta-class is predicted, all the classes belonging to it get a vote. Thus a test example can be easily correctly predicted by at least once. However, RI, Bagging, AdaBoost and other methods hardly achieve such high oracle accuracy, which means that all their trained base learners failed to classify a portion of the test examples. Thus there is still room for those methods to improve by making more diverse predictions.

Evaluation on class imbalanced tasks As the examples in minority classes are fewer in number than those in majority classes, a normal training process which treats all examples equally is often biased toward predicting the majority classes labels. In case the minority classes are regarded as important, a lot of solutions (Chawla 2009) are proposed to somehow balance the influences of the examples in minority classes and the examples in majority classes when training a classifier. Among them, over-sampling the examples in minority classes and down-sampling the examples in majority classes so that the examples in each class are roughly balanced are two fundamental solutions. Their primary goal is to train a single model that exhibits a high classification accuracy.

Ensemble learning, in which no sampling is conducted, is another solution for achieving a high classification accuracy. Note that a large amount of training data is one of the main factors that contribute to the success of deep learning (Goodfellow et al. 2016). Thus down-sampling is not very attractive for training DNNs. We test over-sampling and non-sampling for the compared ensemble learning methods.

Precision and *recall* (Buckland and Gey 1994) are frequently used to evaluate the performance of a classifier trained on a class imbalanced dataset. The precision and recall are defined as $precision = TP/(TP + FP)$ and $recall = TP/(TP + FN)$, where TP is the number of positive examples correctly classified, FP is the number of negative examples classified as positive, and FN is the number of positive examples classified as negative. F-measure (Van Rijsbergen 1979), which is defined as $2 \times precision \times recall/(precision + recall)$, conveys the balance between precision and recall. Note that F-measure, precision and recall are specific to each class, e.g., we have N_C F-measures for an N_C-class classification problem.

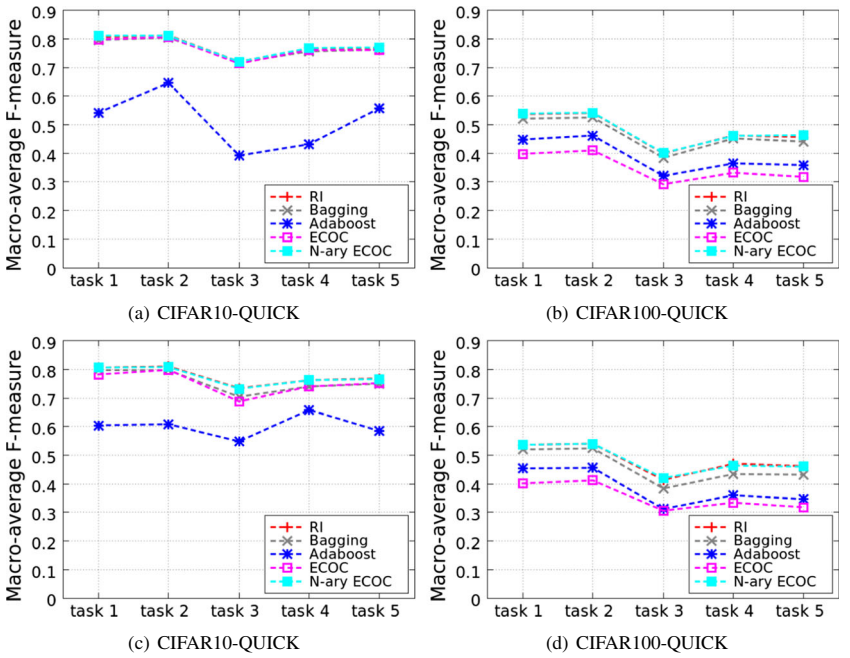
Micro-average and *macro-average* enable evaluating the overall F-measure of all classes, by taking the F-measure of each class into account. Micro-average is obtained by summing up the TPs, FPs and FNs of all classes, and then estimating the overall precision, recall and F-measure. Macro-average is obtained by estimating the overall F-measure through averaging the F-measure of each class, i.e., $(\sum_{i=0}^{N_C-1} F_i)/N_C$, where F_i is the F-measure of class c_i. Thus macro-average, which gives an equal weight to the classification of each class label, is more appropriate for evaluating the class imbalanced tasks. We adopt the macro-average F-measure to evaluate different methods.

We test two DNNs for the class imbalanced tasks, CIFAR10-QUICK and CIFAR100-QUICK. We chose them because they are relatively time-efficient to train. The results are shown in Fig. 1. We see that N-ary ECOC and RI generally obtain competitive performances, and outperform other methods. The low F-measures of Bagging are probably due to the loss of unique data as we discussed previously. Moreover, Bagging and AdaBoost are

Table 5 Oracle accuracies with their standard deviations in the K-fold cross validation

Model	SM	Oracle Accuracy					
		RI	Bagging	AdaBoost	ECOC	N-ary ECOC	Other
MNIST-LeNet$_{\times 60}$	98.93$_{\pm 0.12}$	99.65$_{\pm 0.05}$	99.78$_{\pm 0.03}$	99.97$_{\pm 0.01}$	**100.00**$_{\pm 0.00}$	**100.00**$_{\pm 0.00}$	
SVHN$_{\times 100}$	90.74$_{\pm 0.12}$	98.19$_{\pm 0.15}$	98.57$_{\pm 0.11}$	99.85$_{\pm 0.03}$	**100.00**$_{\pm 0.00}$	**100.00**$_{\pm 0.00}$	
CIFAR10-QUICK$_{\times 100}$	76.31$_{\pm 0.72}$	98.45$_{\pm 0.12}$	98.68$_{\pm 0.11}$	99.82$_{\pm 0.05}$	**100.00**$_{\pm 0.00}$	**100.00**$_{\pm 0.01}$	93.10 (Lee et al. 2015)
CIFAR10-FULL$_{\times 100}$	81.39$_{\pm 0.65}$	97.33$_{\pm 0.14}$	98.04$_{\pm 0.19}$	94.72$_{\pm 3.62}$	**100.00**$_{\pm 0.00}$	**100.00**$_{\pm 0.00}$	
CIFAR100-NIN$_{\times 60}$	59.36$_{\pm 0.30}$	92.61$_{\pm 0.13}$	93.38$_{\pm 0.13}$	93.17$_{\pm 1.88}$	**100.00**$_{\pm 0.00}$	93.50$_{\pm 0.10}$	78.63 (Lee et al. 2015)
CIFAR100-QUICK$_{\times 100}$	44.38$_{\pm 0.29}$	92.21$_{\pm 0.16}$	93.36$_{\pm 0.17}$	94.99$_{\pm 0.18}$	**100.00**$_{\pm 0.00}$	93.53$_{\pm 0.11}$	
FLOWER102-AlexNet$_{\times 60}$	80.89$_{\pm 0.31}$	94.57$_{\pm 0.84}$	96.18$_{\pm 1.08}$	94.57$_{\pm 0.84}$	**100.00**$_{\pm 0.00}$	95.01$_{\pm 0.60}$	

Here N = 3, 3, 7, 4, 95, 95, 95 for the tested DNN models



(a) CIFAR10-QUICK (b) CIFAR100-QUICK

(c) CIFAR10-QUICK (d) CIFAR100-QUICK

Fig. 1 Macro-average F-measures of different methods. Note that the plots in the first row are the results of over-sampling, and the plots in the second row are the results of non-sampling

biased toward maximizing classification accuracy, not F-measure. Note that the F-measures of ECOC are very low when the number of classes is large (CIFAR100-QUICK). This fact is probably due to the low discriminative ability of the ECOC codes.

4.2 Issue 2: How to choose the two crucial parameters (N and N_L) for N-ary ECOC?

General N-ary ECOC In the previous work of N-ary ECOC (Zhou et al. 2016), N represents a value in the range of $3 \leq N < N_C$. In a more general view, we can compare ECOC, N-ary ECOC and RI in a single framework, as they are essentially equivalent in the sense that all the base learners are trained based on the codeword matrix (see Table 1 for examples). We call this framework "general N-ary ECOC", where $N = 2$ represents ECOC, $3 \leq N < N_C$ represents the original N-ary ECOC, and $N = N_C$ represents RI. General N-ary ECOC consists of three procedures: 1) random weight initialization, 2) base learner training based on the codeword matrix and 3) combined voting of the trained base learners. The three methods (ECOC, the original N-ary ECOC and RI) differ in their discriminative abilities of their base learners in terms of their output codes.

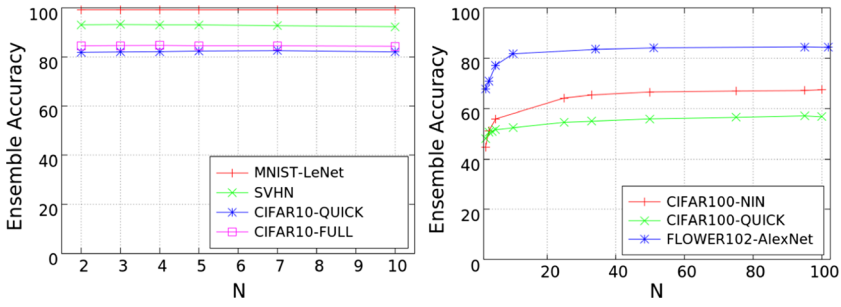
The main reason to compare the three methods in a single framework is that ECOC and RI are well studied and often used in ensemble learning tasks. They help to give a guidance for investigating the behavior of the original N-ary ECOC. In the rest of this Section, we compare the EA of the general N-ary ECOC to gain a deeper understanding of how N and N_L affect the EA.

EA with respect to N Figure 2 shows the EAs of the tested DNN models with respect to N. We see two interesting patterns. *Pattern 1*): the variation of the EA is small. The EA is either almost stable (MNIST-LeNet, SVHN and CIFAR10-FULL), or it increases a *small* amount when N increases at the beginning, and then= begins to decrease a *small* amount after it achieves its optimal value (CIFAR10-QUICK). For both cases, the optimal values for N lie in $3 \leq N < N_C$. This pattern is consistent with the one reported in the previous work of N-ary ECOC (Zhou et al. 2016). The first four models (MNIST-LeNet, SVHN, CIFAR10-QUICK and CIFAR10-FULL) follow this pattern. All the datasets that follow this pattern have a small value for N_C. *Pattern 2*): the EA increases a *large* amount when N increases at the beginning, then it saturates as N continues to increase and approaches N_C. The optimal EA is achieved at N_C or close to N_C. This pattern has not been reported in the previous work of N-ary ECOC (Zhou et al. 2016), which used SVM or decision tree as base learners. The last three models (CIFAR100-NIN, CIFAR100-QUICK and FLOWER102-AlexNet) follow this pattern. All the datasets that follow this pattern have a relatively large value for N_C.

We propose the following *hypothesis* to explain the possible reasons for the observed two patterns. *Hypothesis 1*: The EA is influenced by 1) the quality of the codeword matrix, 2) the number of meta-classes and 3) the degree of merging classes. Among the three factors, the degree of merging classes is more important than the other two because it affects the feature learning. (Case 1) When the number of classes is small, a small value for N is expected to result in a higher EA because the effect of merging classes is smaller than in Case 2. (Case 2) When the number of classes is large, a large value for N is expected to result in a higher EA, as the effect of merging classes is large with a small value for N.

Zhou et al. (2016) identified the effect of the first two factors. That is, the larger the value of N, the better quality of the codeword matrix, thus the higher the EA; the larger the value of N, the larger number of meta-classes, the more difficult (measured by the test error) the classification tasks of the base learners, and thus the lower the EA.

We identified the effect of the third factor. That is, merging classes into meta-classes essentially causes the DNN base learners to learn less representative features, compared with training DNNs with the original class labels (no merging classes). This is mainly due to the fact that there is no need to discriminate the classes within a meta-class for the former. The less representative features, in turn, cause the generalization ability (measured by the test accuracy) of the DNN base learners of N-ary ECOC to decrease. Note that the generalization ability measures how well a trained model generalizes to unseen examples. As we stated previously in Section 4.1, we use $N_C - N$ to measure the degree of merging classes.



(a) Pattern 1 (b) Pattern 2

Fig. 2 Ensemble accuracies with respect to N

The larger the value of $N_C - N$, the greater the effect of merging classes on the learned features and the generalization ability of the trained DNN base learners. Supporting evidence for the effect of the third factor is provided in Appendix A.

Note that the training processes of the base learners of N-ary ECOC for traditional classification tasks (e.g., using SVM or decision tree as base learners) also involve merging classes into meta-classes. However, as they do not need to learn features during training, they do not have the same effect as the DNN base learners have on the final generalization ability of the trained base learners.

We come back to the question of what values for N we should use for N-ary ECOC to achieve its optimal EA. The above experiments suggest that a value for N in the range of 3 to 10 generally exhibits a high EA for the DNNs that follow Pattern 1, and a large value for N is preferred for the DNNs that follow Pattern 2. However, there are two problems related to the above strategy of deciding the optimal value for N. First, training DNNs are generally computationally expensive, thus intensively evaluating the EAs of different values for N is not very attractive. Second, other patterns of EA may be observed in new datasets.

Even with these two problems, the effect of merging classes to the feature learning and the generalization ability of the DNN base learners is a dominating factor for the performance of EA. Thus we can still have a good sense of which values for N are likely to exhibit a high EA by checking the average base model accuracy (BA). A validation procedure regarding this new strategy is proposed at the end of this Section.

In Section 4.1, we observed that N-ary ECOC is more effective when N_C is small (i.e., $N_C < 25$). One possible reason is that, when the number of classes is small, the degree of merging classes (measured by $N_C - N$) is small, thus merging classes has a relatively small effect on the learning process of representative features and the generalization ability of the DNN base learners. Thus, the increased diversity (compared with other ensemble learning methods) is more likely to outweigh the decreased classification accuracy of the DNN base learners of N-ary ECOC for a classification task.

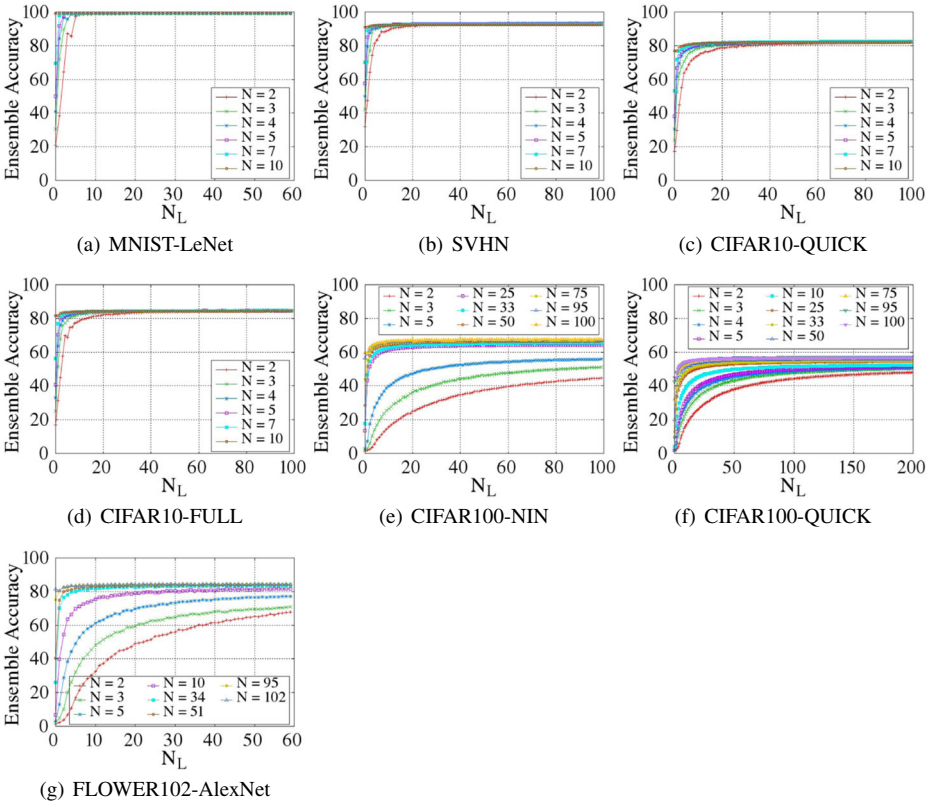
EA with respect to N_L Here we first identify what factors affect the optimal value for N_L in the framework of the general N-ary ECOC. Obviously, the optimal value for N_L is related to N. For a multi-class problem, a small value for N requires a relatively large value for N_L compared with a large value for N, as the discriminative ability of the codes for a small value for N is relatively low. Note that the discriminative ability is measured by how many meta-classes the base learner aims to discriminate. For instance, ECOC ($N = 2$) can only predict 0 or 1 for a given test example, while RI ($N = N_C$) can predict one of the N_C classes.

As the discriminative ability of the codes of N-ary ECOC lies in between that of RI and ECOC, the codeword length needed for N-ary ECOC lies in between the codeword length needed for RI and ECOC. That is,

$$N_L^{opt}(RI) < N_L^{opt}(N\text{-ary } ECOC) < N_L^{opt}(ECOC), \tag{3}$$

where $N_L^{opt}(RI)$, $N_L^{opt}(N\text{-ary } ECOC)$ and $N_L^{opt}(ECOC)$ represent the optimal values for N_L in RI, N-ary ECOC and ECOC, respectively.

Figure 3 shows the EAs of different values for N with respect to N_L for the tested DNN models. We see that the EA of a relatively large value for N tends to converge faster than that of a relatively small value for N. For instance, we see from Fig. 3f that the EA of $N = 100$ converges at around 30, the EA of $N = 10$ converges at around 100, while the EA of $N =$



(a) MNIST-LeNet (b) SVHN (c) CIFAR10-QUICK

(d) CIFAR10-FULL (e) CIFAR100-NIN (f) CIFAR100-QUICK

(g) FLOWER102-AlexNet

Fig. 3 Ensemble accuracies of different values for N with respect to N_L

3 converges at around 200. Thus when N approaches N_C, $N_L^{opt}(N\text{-ary } ECOC)$ is close to $N_L^{opt}(RI)$, and when N approaches 2, $N_L^{opt}(N\text{-ary } ECOC)$ is close to $N_L^{opt}(ECOC)$.

We observed that using 30 base learners for RI is generally good enough whatever N_C is large (e.g., 100) or small (e.g., 10). This pattern can also be seen in Fig. 3. Allwein et al. (2000) empirically showed that, for traditional classifiers, ECOC requires $10\log_2(N_C)$ base learners if we randomly split the classes into two subsets. However, the above conclusion is doubtful for DNNs when N_C is large (e.g., 100). For instance, we see from Fig. 3e, f, g that the EAs of ECOC ($N = 2$ in the plots) tend to increase a large amount when N continues to increase after $10\log_2(N_C)$. Nevertheless, our observations on the $N_L^{opt}(RI)$ and the discussions of Allwein et al. (2000) on $N_L^{opt}(ECOC)$ give us a rough estimation, that is,

$$30 < N_L^{opt}(N\text{-ary } ECOC) < 10\log_2(N_C). \tag{4}$$

If $N_C = 10$, then $10\log_2(N_C) \approx 33$; if $N_C = 100$, then $10\log_2(N_C) \approx 66$. Thus, N-ary ECOC requires approximately no more than 33 and 66 models for DNNs with 10 classes (MNIST-LeNet, SVHN, CIFAR10-QUICK and CIFAR10-FULL) and 100 classes (CIFAR100-QUICK), respectively.

Table 6 lists the EAs of N-ary ECOC with respect to N_L for the tested DNNs. We see that the EAs continue to increase a small amount if we further increase N_L. For instance, the EAs of SVHN, CIFAR10-QUICK and CIFAR10-FULL increase 0.41%, 0.61% and

Table 6 Ensemble accuracies of N-ary ECOC with respect to N_L

Model	Ensemble Accuracy				
	×30	×45	×60	×100	×200
MNIST-LeNet	99.26	**99.27**	**99.27**	–	–
SVHN	92.81	93.03	93.09	**93.22**	–
CIFAR10-QUICK	81.97	82.24	82.31	**82.58**	–
CIFAR10-FULL	84.26	84.44	84.62	**84.68**	–
CIFAR100-NIN	66.76	66.97	67.05	**67.09**	–
CIFAR100-QUICK	56.05	56.51	56.68	56.95	**57.07**
FLOWER102-AlexNet	84.33	**84.47**	**84.47**	–	–

Here N = 3, 3, 7, 4, 95, 95, 95 for the tested DNN models, and "×30" means $N_L = 30$

0.42%, respectively, for $N_L = 100$ compared with $N_L = 30$; the EA of CIFAR100-QUICK increases 0.39% for $N_L = 200$ compared with $N_L = 60$. The above observations show that (4) is a relatively reasonable estimation.

Validation procedure to decide the values of N and N_L (Proposal 1) Here we suggest the following procedure (using a validation data) to decide the optimal values for N and N_L. The overall workflow is shown in Fig. 4. First, we check how the generalization ability of the trained base learners of N-ary ECOC varies with respect to N. Here we can test values, for instance, $N = [3,\ N_C/20,\ N_C/10,\ N_C/5,\ N_C/3,\ N_C/2,\ 3N_C/5,\ 4N_C/5,\ 9N_C/10,\ N_C]$, using $N_L = 1$. If the generalization ability decreases a large amount such that the BA with $N < N_C$ is much lower (e.g., +5%) than the BA of $N = N_C$, a large

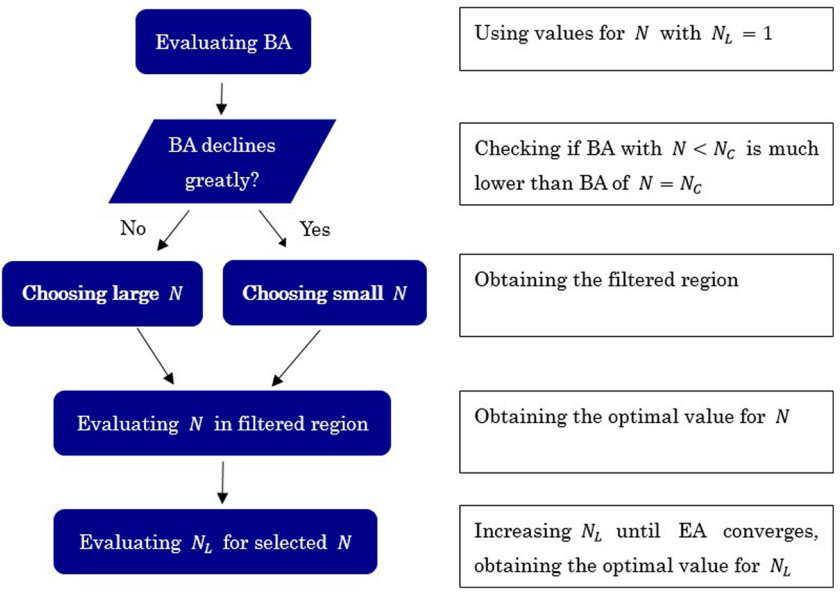


Fig. 4 Validation procedure to decide the values of N and N_L for N-ary ECOC

value for N (e.g., $N > 4N_C/5$) is expected to exhibit a high EA. Otherwise, a small value for N (e.g., $3 \leq N < N_C/2$) is preferred. Second, we check the EAs of different values for N in the filtered region (e.g., $N > 4N_C/5$ or $3 \leq N < N_C/2$) in the first step, and decide the optimal value for N (e.g., $N = 90$). Here we can use a relatively large value for N_L, e.g., $N_L = 10\log_2(N_C)$. Third, for the selected value for N (e.g., $N = 90$), we increase N_L until EA converges.

Suppose we investigate M_1 different values for N in the first step with $N_L = 1$, and M_2 different values for N in the second step with $N_L = 10\log_2(N_C)$, then the overall training time is $T(M_1 + M_2 10\log_2(N_C))$, where T is the training time for a single DNN. Note that if we directly investigate the EAs of $M_1 + M_2$ DNNs with $N_L = 10\log_2(N_C)$, the computational cost is $T(M_1 + M_2)10\log_2(N_C)$.

Our proposed validation procedure contributes to a more efficient setting of the two hyperparameters of N-ary ECOC. It would possibly help deep learning practitioners to avoid wasting weeks or months by waiting for the training of DNNs to tune these two hyperparameters. For instance, training a single CIFAR100-NIN network takes approximately 5.5 hours based on our experiments, and thus it takes approximately 137 days to evaluate 10 different values for N if we train 60 base models for each N. With our proposal, the time can be reduced to a few days.

More importantly, our proposal can help to avoid searching an optimal value for N in a totally wrong region. For instance, we may search a value for N in a range from 3 to 10 (as suggested by the previous work of N-ary ECOC (Zhou et al. 2016)) when an optimal value is actually around 90 for a 100-class classification problem.

4.3 Issue 3: Can we use RI or HV to improve the ensemble accuracy of N-ary ECOC?

N-ary ECOC makes an ensemble of base learners trained with codes f_0, ..., f_{N_L-1} (see Table 1 (b) for an example). RI or HV can serve as a preprocessing step for N-ary ECOC to enhance the base learner trained with a single code. For instance, in the case of RI, we train ν DNNs with different initializations using the code f_n, and then make the combined votes $f_n^{RI} = majority\_voting(f_n^0, ..., f_n^{\nu-1})$ as the predictions of the DNN trained with f_n. As we stated previously in Section 1, HV aggregates a single DNN at different iterations after the network is stable (Xie et al. 2013). Thus in the case of HV, we train a single DNN using the code f_n but save ν snapshots of its parameters after the DNN is stable, and then make the combined votes $f_n^{HV} = majority\_voting(f_n^{iter_0}, ..., f_n^{iter_{\nu-1}})$. Hence both RI and HV can improve the BA of the base learner of N-ary ECOC trained with f_n by training a number of them. It is natural to ask if RI or HV further improves the overall EA of N-ary ECOC.

To verify this hypothesis, we conducted experiments on MNIST-LeNet, SVHN and CIFAR10-QUICK, because they are relatively time-efficient to train compared with the other models. Note that if $N_L = 100$ and $\nu = 15$, we need to train 1,500 DNNs for RI enhanced N-ary ECOC. Though the three datasets involved (MNIST, SVHN and CIFAR10) all have a small value for N_C (i.e., $N_C = 10$), we believe the conclusions driven from them also hold true for DNNs with a large value for N_C (e.g., $N_C \geq 100$), as RI or HV generally makes better predictions than a single model no matter if N_C is large or small. We use N = 3 for the three DNN models because fixing the value of N makes it easy for comparing different methods. Note that we could choose other values for N, for instance, N = 4 or 5.

Table 7 shows the results. Here N-ary ECOC + RI (N-ary ECOC combined with RI) trains 15 individual models (i.e., $\nu = 15$) for each code f_n. N-ary ECOC + HV (N-ary ECOC combined with HV) runs additional 2,000 iterations and saves a snapshot for every

Table 7 Ensemble performances of N-ary ECOC, N-ary ECOC + RI (N-ary ECOC combined with RI) and N-ary ECOC + HV (N-ary ECOC combined with HV)

Model	EA	BA	Q_{avg}	Dis_{avg}
(a) N-ary ECOC				
MNIST-LeNet$_{\times 60}$	99.27	99.10$_{\pm 0.14}$	**0.974**	**0.355**
SVHN$_{\times 100}$	93.13	92.43$_{\pm 0.84}$	**0.854**	**0.363**
CIFAR10-QUICK$_{\times 100}$	82.16	80.95$_{\pm 2.40}$	**0.637**	**0.380**
(b) N-ary ECOC + RI				
MNIST-LeNet$_{\times 60}$	**99.28**	**99.21**$_{\pm 0.12}$	0.981	0.354
SVHN$_{\times 100}$	93.08	**93.28**$_{\pm 0.75}$	0.889	0.360
CIFAR10-QUICK$_{\times 100}$	**82.17**	**84.13**$_{\pm 2.03}$	0.756	0.370
(c) N-ary ECOC + HV				
MNIST-LeNet$_{\times 60}$	99.25	**99.21**$_{\pm 0.12}$	0.979	0.354
SVHN$_{\times 100}$	**93.24**	92.63$_{\pm 0.83}$	0.860	0.362
CIFAR10-QUICK$_{\times 100}$	82.08	81.14$_{\pm 2.34}$	0.644	0.379

Here $N = 3$ for all three methods. Q_{avg} (the smaller the value, the more diverse the predictions) and Dis_{avg} (the larger the value, the more diverse the predictions) are two diversity measures

200 iterations. We see from Table 7 that the EAs of N-ary ECOC + RI and N-ary ECOC + HV are almost equal to the EA of N-ary ECOC for each DNN.

As discussed in Section 4.1, EA is mainly influenced by the accuracy and diversity of the base learners. To investigate the reason why RI and HV failed to improve the overall EA of N-ary ECOC, we list the BA and estimate the diversity of the trained base learners of different methods in Table 7. Besides the Q statistic (Yule 1900) measure, we employ another often used diversity measure, the disagreement measure (Skalak 1996), to evaluate the diversity. With the two diversity measures, we can obtain a more robust estimation.

The disagreement measure is for estimating the difference between the predictions of two classifiers C_i and C_j by

$$Dis_{i,j} = \frac{1}{m} \sum_{k=1}^{m} \mathbb{1}(y_{k,i} = y_{k,j}), \tag{5}$$

where m is the number of examples in the test dataset, $y_{k,i}$ and $y_{k,j}$ are the predictions of C_i and C_j, respectively, on the kth test example. $\mathbb{1}(y_{k,i} = y_{k,j})$ is an indicator function, which outputs 1 if $y_{k,i}$ and $y_{k,j}$ are equal, and 0 otherwise.

One main advantage of the disagreement measure over Q statistic is that it does not need to access the class labels. This advantage is very useful when class labels are not (fully) available, e.g., semi-supervised learning. However, (5) is not directly applicable to the base learners of N-ary ECOC because a meta-class of different base learners with the same identifier generally consists of different classes. Take Table 1b as an example, the meta-class 0 of f_0 consists of the classes c_0 and c_8, while the meta-class 0 of f_2 consists of the classes c_2 and c_6, and thus directly measuring the differences of the predicted meta-class labels of two classifiers with (5) does not make sense. To deal with this problem, we extended the disagreement measure as shown in Appendix B. Note that the values for Dis_{avg} in Table 7 are based on our extended disagreement measure.

We see from Table 7 that both N-ary ECOC + RI and N-ary ECOC+ HV improve the BA of N-ary ECOC. Note that N-ary ECOC + RI even improves the BA of N-ary ECOC a large amount (from 80.95% to 84.13%) for CIFAR10-QUICK. However, the trained base learners of N-ary ECOC are always more diverse than those of N-ary ECOC + RI and N-ary ECOC + HV, which suggests that the RI and HV methods cause the base learners of N-ary ECOC to be less diverse. Note that the BA, Q_{avg} and Dis_{avg} of N-ary ECOC + RI and N-ary ECOC + HV are all measured based on f_n^{RI} or f_n^{HV} (after majority voting). For the diversities, the possible reason is that, by the preprocessing of RI or HV, some errors are corrected. Therefore, the predictions of the base learners tend to be relatively similar compared with no preprocessing step performed.

Through the above experiments, we see that N-ary ECOC combined with RI or HV does not increase the EA of N-ary ECOC, though each of them may improve the BA of N-ary ECOC.

5 Conclusion

In this paper, we investigated three important issues regarding N-ary ECOC with DNNs as base learners. 1) Which ensemble learning method to use? 2) How to choose the two crucial parameters (N and N_L) for N-ary ECOC? 3) Can we use RI or HV to improve the ensemble accuracy of N-ary ECOC?

For the first issue, we found that N-ary ECOC generally exhibits a superior ensemble accuracy compared with RI, ECOC, Bagging, AdaBoost and two other ensemble learning methods, especially when the number of classes is small (i.e., $N_C < 25$). Interesting applications include image classification tasks with a small number of classes such as optical character recognition.

For the second issue, our experiments suggest that EA is largely influenced by the effect of merging classes. When the decrease of the generalization ability (measured by the BA) caused by merging classes is large, using a large value for N generally results in a high EA. Otherwise, using a small value for N is expected to result in a high EA. We found that an optimal value for N_L roughly lies in the range of $(30, 10\log_2(N_C))$. We proposed a validation procedure to evaluate the optimal values for N and N_L.

An important finding related to Issue 2 is that merging classes generally causes the DNN base learners of N-ary ECOC to learn less representative features, which results in the decrease of the generalization ability, compared with training DNNs using the original class labels. An interesting application of this finding is to train DNNs using the sub-class labels instead of the original class labels if we know that there exist sub-classes in the data and know the corresponding sub-class labels. Here we only assume that there is at least one class that contains sub-classes, and not necessarily that a class hierarchy exists in the classes. The use of the sub-class labels instead of the original class labels is likely to avoid the decrease of the generalization ability caused by the merging of sub-classes.

To overcome the effect of merging classes on the feature learning for the base learners of N-ary ECOC, we propose to use the original class labels in the first half of the training process, and then use the meta-class labels in the second half of the training process. Our future work will explore this direction.

For the third issue, we found that using the previous ensemble learning methods RI and HV to enhance the single base learner of N-ary ECOC did not improve the overall ensemble accuracy. The reason is that RI and HV decrease the diversity of the base learners of N-ary

ECOC. Our work can be viewed as a test of the strategy of improving the BA at the price of decreasing the diversity of the base learners. However, the opposite strategy, i.e., improving the diversity of the base learners at the price of scarifying BA a small amount, remains an interesting direction. This is because an ensemble of relatively diverse predictions tends to correct more errors. Our finding can help future researchers to avoid wasting time on the idea (and possibly the extension of the idea) of using an existing ensemble learning method such as RI and HV to enhance the single base learner of N-ary ECOC and invent new ensemble learning methods along the opposite strategy.

We are aware that the above findings are only driven by the experiments of deep learning of image classification tasks. Whether these conclusions are applicable to other deep learning tasks such as speech recognition and natural language processing remains unexplored.

Other contributions of this paper are: 1) we provided a unified framework called "general N-ary ECOC" for the ECOC, N-ary ECOC and RI methods. The new framework enables us to choose the value for N from a wide range (i.e., $2 \leq N \leq N_C$) for ensemble learning of DNNs, as the base learners of the three methods are all trained based on a codeword matrix; and 2) we extended the previous disagreement measure (Skalak 1996). This extension enables us to evaluate the diversity of the predictions of the base learners of ECOC and N-ary ECOC without accessing the class labels, which is useful when class labels are not (fully) available such as semi-supervised learning.

Appendix

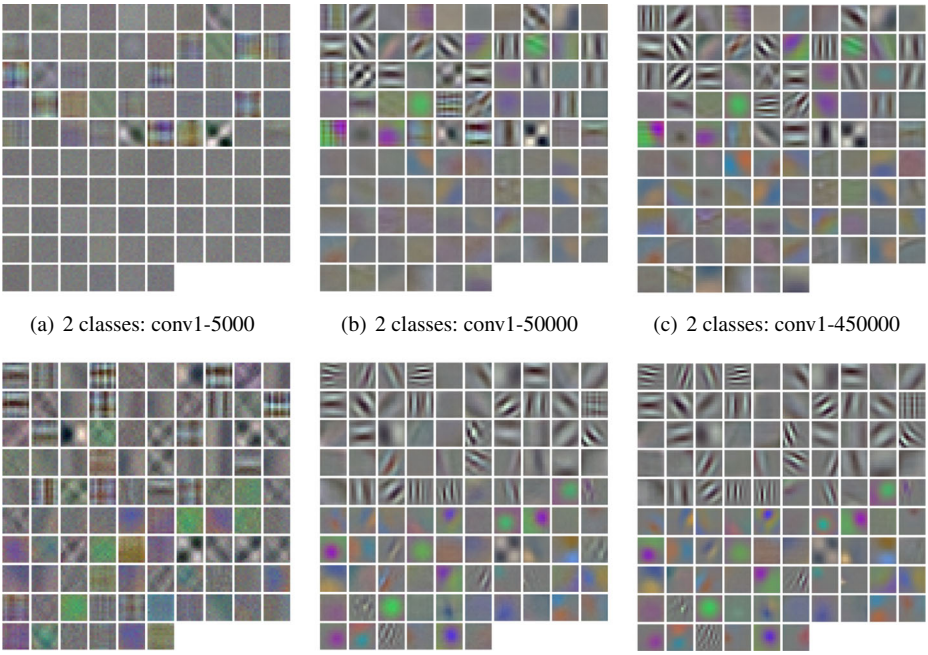
A. Supporting Evidences for the Effect of the Third Factor in *Hypothesis 1*

To verify the effect of merging classes on the feature learning of the DNN base learners of N-ary ECOC, we train the AlexNet (Krizhevsky et al. 2012) using the ILSVRC2012 dataset with two different encoding strategies. The first one is ECOC encoding, in which the degree of merging classes is large. We follow Yosinski et al. (2014) to split the 1,000 classes into two meta-classes. Each meta-class contains approximately half (551 classes versus 449 classes) of the 1,000 original classes. The second one is RI encoding (keeping the original 1,000 classes), in which the degree of merging classes is small.

Figure 5 shows the visualizations of the learned features of the first convolutional layer at different iterations (5,000, 50,000 and 450,000) for the two encoding strategies. There are six plots, in which Figs. 5a, b, c and d, e, f correspond to ECOC and RI encodings, respectively. For each plot, there are 96 squared blocks, each of which represents a learned *feature map* (also called *filter*) of size 11 (width) $\times$11 (height) $\times$3 (channel).

We see from Fig. 5 that a DNN with two meta-classes (Fig. 5a, b, c) tends to learn less representative features compared with a DNN with 1,000 classes (Fig. 5d, e and f). For instance, if we compare Fig. 5a with Fig. 5d, we see that most of the learned features of the DNN with ECOC encoding at the 5,000th iteration look like noisy features (e.g., the lower half of the rows of the learned features in Fig. 5a). However, the DNN with RI encoding at the 5,000th iteration has learned many representative features (e.g., the edges and the color patterns in Fig. 5d).

If we compare Fig. 5b with Fig. 5e, i.e., at the 50,000th iteration, we see that the DNN with ECOC encoding (Fig. 5b) has learned more representative features (compared with itself at the 5,000th iteration in Fig. 5a), while the DNN with RI encoding (Fig. 5e) keeps refining its learned features.



(a) 2 classes: conv1-5000 (b) 2 classes: conv1-50000 (c) 2 classes: conv1-450000

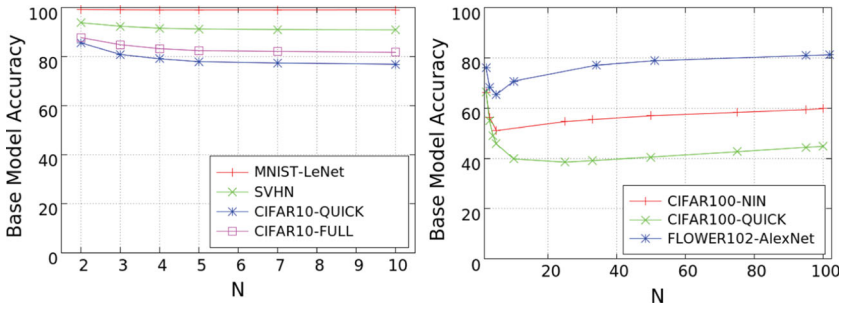
(d) 1000 classes: conv1-5000 (e) 1000 classes: conv1-50000 (f) 1000 classes: conv1-450000

Fig. 5 Visualizations of the first convolutional layer (*conv1*) of the trained AlexNet models with two meta-classes (the first row) and 1,000 meta-classes (the second row) at different iterations. Note that the maximum number of iterations is 450,000

If we compare Fig. 5c with Fig. 5f, i.e., at the 450,000th iteration (the end of the training), we see that the learned features of the DNN with ECOC encoding (Fig. 5c) are not as representative as those of the DNN with RI encoding (Fig. 5f) in terms of the complexity of the patterns of the learned features. For instance, most of the color patches of the DNN with ECOC encoding represent some kind of gradient maps to the edge direction, while many of the color patches of the DNN with RI encoding represent more complex gradient maps such as a color patch surrounded by other different color patches (e.g., the features in the 6th row and the 5th column, the 6th row and the 7th column, and the last row and the 5th column of Fig. 5f), which are useful in detecting a sharp change in color in almost all directions.

To verify the effect of merging classes on the generalization ability of the DNN base learners of N-ary ECOC, we show the average base model accuracies (BAs) of the tested DNN models (as listed in Table 2 in Section 3.1) with respect to N in Fig. 6, as BA shows how well the learned DNNs generalize to the test data. We see from Fig. 6 that, for the DNNs that follow Pattern 1 (MNIST-LeNet, SVHN, CIFAR10-QUICK and CIFAR10-FULL)), the BA generally decreases as N increases. Moreover, the base learners with $N < N_C$ have BAs which are higher than the BA of RI ($N = N_C$). Two possible reasons behind the above observations are, for a dataset with a small number of classes (i.e., $N_C < 25$), the effect of merging classes is small, and the BA is practically determined by the value of N (the smaller N, the easier the classification task).

On the other hand, for the DNNs that follow Pattern 2 (CIFAR100-NIN, CIFAR100-QUICK and FLOWER102-AlexNet), the BA first decreases a large amount (more than



(a) DNNs that follow Pattern 1 (b) DNNs that follow Pattern 2

Fig. 6 Average base model accuracies with respect to N

10%), then slowly increases as N continues to increase. Moreover, most of the base learners with $N < N_C$ have BAs which are much lower than the BA of RI. Intuitively, an N-class ($N < 100$) classification problem is easier than a 100-class classification problem. However, the results are the opposite. One possible reason behind the above observations is that, for a dataset with a large number of classes (e.g., $N_C \geq 100$), when N is small ($N \ll N_C$), $N_C - N$ is large (i.e., the degree of merging classes is large), thus the BA is generally low compared with the BA of RI. Note that the DNN with $N = 2$ (a binary classification problem) corresponds to the easiest classification task (i.e., a random guess will be 50% correct if the class distribution is balanced) among all possible values for N ($2 \leq N \leq N_C$), and thus it can still have a high BA. When N begins to increase, the accuracy of a random guess drops sharply, for instance, 33% for $N = 3$ and 25% for $N = 4$, even if the class distribution is balanced. Thus there is a drop for the BA. When N further increases, the effect of merging classes is weakened, thus the BA begins to increase and finally saturates.

The above observations suggest that both the feature learning and the generalization ability of the DNN base learners of N-ary ECOC are largely influenced by the effect of merging classes.

B. Our extended disagreement measure

As defined in Section 4.3, m is the number of examples in the test dataset, C_i and C_j are two base learners of N-ary ECOC, $y_{k,i}$ and $y_{k,j}$ are the predictions of C_i and C_j, respectively, on the kth test example. We propose to map the predicted meta-class label $y_{k,i}$ (each label takes a value from $0, \ldots, N-1$) to the original class label (each label takes a value from $0, \ldots, N_C - 1$), and then construct a one-by-N_C bit vector $y'_{k,i}$ to indicate if the original class labels are included in the predicted meta-class $y_{k,i}$. Thus the mapping from $y_{k,i}$ to $y'_{k,i}$ is $\{0, \ldots, N-1\} \rightarrow \{0, \ldots, N_C - 1\}^{N_C}$.

Take Table 1 (b) in Section 2 as an example, suppose the base learners trained with f_0 and f_2 both predict the meta-class 0, for the kth test example, i.e., $y_{k,0} = 0$ and $y_{k,2} = 0$. Since the meta-class 0 of f_0 consists of the classes c_0 and c_8 while the meta-class 0 of f_2 consists of the classes c_2 and c_6, the corresponding $y'_{k,0}$ and $y'_{k,2}$ are $y'_{k,0}$ = [1 0 0 0 0 0 0 0 1 0] and $y'_{k,2}$ = [0 0 1 0 0 0 1 0 0 0]. That is, the c_0 and c_8 positions of $y'_{k,0}$ as well as the c_2 and c_6 positions of $y'_{k,2}$ are 1, while all other positions are 0.

Then we estimate the Hamming distance between $y'_{k,i}$ and $y'_{k,j}$, i.e., $(y'_{k,i} - y'_{k,j})(y'_{k,i} - y'_{k,j})^T$, as the distance between $y_{k,i}$ and $y_{k,j}$. For instance, in the above example, the distance between the predictions of the meta-class 0 by DNNs trained with f_0 and f_2 is 4 as four bit positions of $y'_{k,i}$ and $y'_{k,j}$ are different.

The advantage of our proposal is that it fits our intuition. That is, the more overlap exists in the classes included in the predicted meta-classes, the smaller the distance. In a general case that the predicted meta-classes of two base learners share common classes, the distance measured by our proposal will be smaller than the above example, in which no common class exists. For instance, if the predicted meta-classes are 2 and 3 for f_0 and f_2, respectively (note that the meta-class 2 of f_0 and the meta-class 3 of f_2 share a common class c_1), then the distance is 2 as only two bit positions of $y'_{k,i}$ and $y'_{k,j}$ are different. In the extreme case that the classes included in the predicted meta-classes are identical, the distance is 0. For instance, both the meta-class 0 of f_0 and the meta-class 4 of f_2 consist of the classes c_0 and c_8.

Finally, we propose to estimate the disagreement measure of the base learners of N-ary ECOC by

$$Dis'_{i,j} = \frac{1}{m}\sum_{k=1}^{m}(y'_{k,i} - y'_{k,j})(y'_{k,i} - y'_{k,j})^T, \tag{6}$$

where $y'_{k,i}$ and $y'_{k,j}$ are the mapped back predictions of the base learners C_i and C_j, respectively, on the kth test example.

By averaging the disagreement measures over all pairs of base learners based on Skalak (1996), we obtain

$$Dis_{avg} = \frac{2}{N_L(N_L-1)}\sum_{i=1}^{N_L-1}\sum_{j=i+1}^{N_L} Dis'_{i,j}. \tag{7}$$

References

Ahmed, E., Jones, M., Marks, T.K. (2015). An improved deep learning architecture for person re-identification. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 3908–3916.

Ahonen, T., Hadid, A., Pietikainen, M. (2006). Face description with local binary patterns: Application to face recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *28*(12), 2037–2041.

Allwein, E.L., Schapire, R.E., Singer, Y. (2000). Reducing multiclass to binary: A unifying approach for margin classifiers. *Journal of Machine Learning Research*, *1*, 113–141.

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

Buckland, M., & Gey, F. (1994). The relationship between recall and precision. *Journal of the American Society for Information Science*, *45*(1), 12.

Buda, M., Maki, A., Mazurowski, M.A. (2017). A systematic study of the class imbalance problem in convolutional neural networks. arXiv:1710.05381.

Chawla, N.V. (2009). Data mining for imbalanced datasets: An overview. In: Data mining and knowledge discovery handbook, pp. 875–886. Springer.

Cortes, C., & Vapnik, V. (1995). Support-vector networks. *Machine Learning*, *20*(3), 273–297.

Dietterich, T.G. (2002). *Ensemble Learning. The handbook of brain theory and neural networks*, 2nd edn. Cambridge: MIT Press.

Dietterich, T.G., & Bakiri, G. (1995). Solving multiclass learning problems via error-correcting output codes. *Journal of Artificial Intelligence Research*, *2*(1), 263–286.

Freund, Y., & Schapire, R.E. (1997). A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, *55*(1), 119–139.

Goodfellow, I., Bengio, Y., Courville, A. (2016). *Deep learning*. London: MIT Press.

Han, J., Pei, J., Kamber, M. (2011). Data mining: Concepts and techniques, 3rd edn., Waltham, Morgan Kaufmann.

Hastie, T., Rosset, S., Zhu, J., Zou, H. (2009). Multi-class AdaBoost. *Statistics and Its Interface*, *2*(3), 349–360.

Ho, T.K. (1995). Random Decision Forests. In: Proceedings of the 3rd international conference on document analysis and recognition, pp. 278–282.

Jia, Y., Shelhamer, E., Donahue, J., Karayev, S., Long, J., Girshick, R., Guadarrama, S., Darrell, T. (2014). Caffe: convolutional architecture for fast feature embedding. In: Proceedings of the 22nd ACM international conference on multimedia, pp. 675–678.

Krizhevsky, A., & Hinton, G. (2009). *Learning multiple layers of features from tiny images. Master's thesis, department of computer science*. Canada: University of Toronto.

Krizhevsky, A., Sutskever, I., Hinton, G.E. (2012). Imagenet classification with deep convolutional neural networks. In: Proceedings of the advances in neural information processing systems, pp. 1097–1105.

LeCun, Y., Bottou, L., Bengio, Y., Haffner, P. (1998). Gradient-based learning applied to document recognition. In: Proceedings of the IEEE, pp. 2278–2324.

Lee, S., Purushwalkam, S., Cogswell, M., Crandall, D., Batra, D. (2015). Why *M* heads are better than one: Training a diverse ensemble of deep networks. arXiv:1511.06314.

Lin, M., Chen, Q., Yan, S. (2014). Network in network. In: Proceedings of the international conference on learning representations.

Masko, D., & Hensman, P. (2015). *The impact of imbalanced training data for convolutional neural networks. Bachelor's thesis, school of computer science and communication*. Sweden: KTH Royal Institute of Technology.

Müller, H., Michoux, N., Bandon, D., Geissbuhler, A. (2004). A review of content-based image retrieval systems in medical applications - clinical benefits and future directions. *International Journal of Medical Informatics*, *73*(1), 1–23.

Netzer, Y., Wang, T., Coates, A., Bissacco, A., Wu, B., Ng, A.Y. (2011). Reading digits in natural images with unsupervised feature learning. In: Proceedings of the NIPS workshop on deep learning and unsupervised feature learning.

Nilsback, M.E., & Zisserman, A. (2008). Automated flower classification over a large number of classes. In: Proceedings of the 6th Indian conference on computer vision, graphics & image processing, pp. 722–729.

Opitz, M., Possegger, H., Bischof, H. (2016). Efficient model averaging for deep neural networks. In: Proceedings of the 13th Asian conference on computer vision (ACCV'16).

Pearson, K. (1895). Note on regression and inheritance in the case of two parents. *Proceedings of the Royal Society of London*, *58*, 240–242.

Quinlan, J.R. (1986). Induction of decision trees. *Machine Learning*, *1*(1), 81–106.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., et al. (2015). Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, *115*(3), 211–252.

Schapire, R.E. (1990). The strength of weak learnability. *Machine learning*, *5*(2), 197–227.

Seiffert, C., Khoshgoftaar, T.M., Van Hulse, J., Napolitano, A. (2008). Resampling or reweighting: A comparison of boosting implementations. In: Proceedings of the IEEE international conference on tools with artificial intelligence, pp. 445–451.

Skalak, D.B. (1996). The sources of increased accuracy for two proposed boosting algorithms. In: Proceedings of the american association for artificial intelligence workshop on integrating multiple learned models.

Van Rijsbergen, C. (1979). *Information Retrieval*. London: Butterworth.

Vriesmann, L.M., de Souza Britto Jr, A., De Oliveira, L.E.S., Sabourin, R., Ko, A.H.R. (2012). Improving a dynamic ensemble selection method based on oracle information. *International Journal of Innovative Computing and Applications*, *4*(3-4), 184–200.

Wei, X.S., Gao, B.B., Wu, J. (2015). Deep spatial pyramid ensemble for cultural event recognition. In: Proceedings of the IEEE international conference on computer vision workshops, pp. 38–44.

Xie, J., Xu, B., Chuang, Z. (2013). Horizontal and vertical ensemble with deep representation for classification. In: Proceedings of the ICML workshop on representation learning.

Yosinski, J., Clune, J., Bengio, Y., Lipson, H. (2014). How transferable are features in deep neural networks?. In: Proceedings of the advances in neural information processing systems, pp. 3320–3328.

Yule, G.U. (1900). On the association of attributes in statistics: With illustrations from the material of the childhood society. *Philosophical Transactions of the Royal Society of London Series A*, *194*, 257–319.

Zhou, J.T., Tsang, I.W., Ho, S.S., Muller, K.R. (2016). *N*-ary error correcting coding scheme. arXiv:1603.05850.

Zhou, Z.H. (2012). Ensemble methods: Foundations and algorithms, Chapman & Hall/CRC, Boca Raton.