Deep *N*-ary Error Correcting Output Codes

Hao Zhang[1,*], Joey Tianyi Zhou[1], Tianying Wang[1], Ivor W. Tsang[2], Rick Siow Mong Goh[1]
{zhang_hao, joey_zhou, wang_tianying, gohsm}@ihpc.a-star.edu.sg[1], ivor.tsang@uts.edu.au[2]

Institute of High Performance Computing, A*STAR[1], AAII, University of Technology Sydney[2]

Abstract. Ensemble learning consistently improves the performance of multi-class classification through aggregating a series of base classifiers. To this end, data-independent ensemble methods like Error Correcting Output Codes (ECOC) attract increasing attention due to its easiness of implementation and parallelization. Specifically, traditional ECOCs and its general extension *N*-ary ECOC decompose the original multi-class classification problem into a series of independent simpler classification subproblems. Unfortunately, integrating ECOCs, especially *N*-ary ECOC with deep neural networks, termed as *Deep N-ary ECOC*, is not straightforward and yet fully exploited in the literature, due to the high expense of training base learners. To facilitate the training of *N*-ary ECOC with deep learning base learners, we further propose three different variants of parameter sharing architectures for deep *N*-ary ECOC. To verify the generalization ability of deep *N*-ary ECOC, we conduct experiments by varying the backbone with different deep neural network architectures for both image and text classification tasks. Furthermore, extensive ablation studies on deep *N*-ary ECOC show its superior performance over other deep data-independent ensemble methods.

Keywords: deep *n*-ary ecoc; ensemble learning; multi-class classification.

1 Introduction

Multi-class classification is one of the fundamental problems in machine learning and data mining communities, where one trains a model with labeled data of different classes for classification purposes. The multi-class classification exists diverse real-world applications from computer vision tasks such as object recognition [1], [2], [3], face verification [4], to natural language processing tasks like sentiment classification [5], [6].

To handle multi-class classification problems, existing approaches could be mainly divided into two groups. One group focuses on solving the multi-class problems directly by extending its corresponding binary classification algorithm. These approaches include decision tree-based methods [7], multi-class linear discriminant analysis [8], multi-layer perceptron [9], multi-class support vector machines (SVM) [10] and etc. Another research direction focuses on the decomposition of a multi-class problem into multiple binary subproblems so that one can reuse the well-studied binary classification algorithms for their simplicity and efficiency. Most of these methods can be reinterpreted in the framework of error correcting output codes (ECOC) [11], [12]. For example, Allwein *et al.* [13] show one-versus-one (OVO), one-versus-all (OVA) could be incorporated into the framework of ECOC

* Corresponding Author

EAI MOBIMEDIA 2020, August 27-28, Harbin, People's Republic of China

DOI 10.4108/eai.27-8-2020.2299197

where all the classes are reassigned with either binary codes $\{-1,1\}$ or ternary codes $\{-1,0,1\}$ for each base learners ($1/-1$ represents positive/negative class, 0 represents non-considered class). Zhou *et al.* [14] further extend traditional ECOCs into *N*-ary ECOC by introducing *N* meta-classes rather than binary classification for each base learner. The final results are determined by the ensemble of a series of base learners. The biggest advantage of ECOCs methods is their easiness of implementation and parallelization.

Most traditional ECOCs methods are based on the pre-defined hand-craft features and focus on how to ensemble the results of base learners on these features. Recently, deep learning methods significantly advance the multi-class classification performance through learning features in an end-to-end fashion. For example, a single AlexNet [15] outperforms the second place at the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) by more than 10%. To further improve performance, Goodfellow *et al.* [16] demonstrate that a simple ensemble of seven AlexNet models with different random initiations could significantly reduce an error rate from 18.2% to 15.3%. In most high profile competitions, e.g. ImageNet[1] [17] or Kaggle[2], ensembles techniques often appear in the winner solution. Traditional ensemble methods usually assume that the base learners for binary classification are inexpensive to train, such as SVMs and decision trees. Unfortunately, this assumption appears to be invalid with deep learning algorithms. For example, AlexNet consisting of more than 60 millions of parameters [15] takes between five and six days to train on two GTX 580 3GB GPUs. Therefore, the expensive learning procedure hinders the use of the ensemble of deep neural networks on a large scale.

In this paper, we focus on addressing the ensemble of deep neural networks in the framework of ECOC. The biggest reason to choose ECOC rather than other ensemble techniques such as Boosting [18] is that ECOC is easy to parallel due to the independence of base learners. In contrast, boosting trains a number of models sequentially and continuously compensates the mistakes made by the earlier models, which results in that each base models in boosting are highly dependent on each other. At this point, ECOC exhibits a large advantage in large-scale real-world applications since all the base learners could be trained independently and simultaneously.

Specifically, we choose *N*-ary ECOC, an extension of ECOC, which shows significant improvement over OVA, OVO, and traditional ECOCs [14]. As known, most existing works did not investigate the influence of deep learning on ECOCs or *N*-ary ECOC. In this paper, we make a marriage between *N*-ary ECOC to investigate such an influence. In the sequence, we term this problem as *Deep N-ary ECOC*. The main contributions of this paper are as follows:

- We investigate a new problem named *Deep N-ary ECOC* where we mainly discuss how to effectively and efficiently leverage advantages of deep learning models in the framework of ECOC.
- To facilitate the training procedure, we further propose three different parameter sharing strategies for Deep *N*-ary ECOC framework, *i.e.*, full parameters share, partial parameters share, and no parameter share. Specifically, the full share model shares all the feature learning parameters except for top classifier; the partial share model shares part of feature learning parameters; the no parameter share means all the base learners are learned from scratch.

[1] ImageNet: http://www.image-net.org/
[2] Kaggle: https://www.kaggle.com/

- We explore the influence of two crucial hyper-parameters of *N*-ary ECOC, *i.e.*, N_L and N, with deep neural networks for improving the accuracy. We also give specific suggestions for choosing those two hyper-parameters.
- We conduct extensive experiments and compare with several ensemble strategies, *i.e.*, an ensemble of random initialization (ERI), ECOC and *N*-ary ECOC, on both image and text classification tasks to analyze the advantages and disadvantages of each ensemble strategy.

The rest of this paper is organized as follows. Section 2 reviews related work. Section 3 presents Deep *N*-ary ECOC. Finally, Section 4 discusses our empirical studies and Section 5 concludes this work.

2 Related Work

Our proposed deep *N*-ary ECOC is highly related to the following topics, including ECOCs, ensemble learning, and deep neural networks.

2.1 ECOCs

Many ECOC approaches [13], [19], [20], [21] have been proposed to design a good coding matrix in recent years. Most of them are fallen into the following two categories. The first one is data-independent coding, such as OVO, OVA, and ECOCs [20]. Their coding matrix design is not optimized for the training dataset nor the instance labels such that all the base learners could be independently learned. For example, the sparse ECOC coding approach aims to construct the ECOC matrix $M \in \{-1, 0, 1\}^{N_C \times N_L}$, where N_C is the number of classes, N_L is the code length, and its elements are randomly chosen as either -1, 1, or 0 [20]. In ECOCs, the classes corresponding to $1, -1$ are considered as positive and negative classes, respectively, and 0 are not considered in the learning process. More recently, Zhou *et al.* [14] extend the existing ECOCs into *N*-ary ECOC to enable the construction of *N* metaclasses. Both theoretical and empirical findings validate the superiority of *N*-ary ECOC over traditional ECOCs.

Another direction is data-dependent ECOCs where the data are considered in the learning coding matrix, such as discriminant ECOC (D-ECOC) [19], ECOC-ONE [22], subspace ECOC [21], Adaptive ECOC [23], etc. In this way, different base learners interact with each other during training phrases, which is also similar to Boosting [18] methods, such as AdaBoost [24]. In Boosting methods, a series of models are sequentially trained with latter models correcting mistakes committed in previous models. Compared to data-independent ECOCs, these methods require sophisticated algorithm design and are difficult to be paralleled.

To our best knowledge, there is little research to investigate the combination of ECOCs and deep learning. In this paper, we take a step further to analyze the performance of combining our previous work *N*-ary ECOC with deep learning.

2.2 Deep Ensemble Learning

A lot of studies show that deep neural network models are nonlinear and have a high variance, which can be frustrating when preparing a final model for making predictions [16].

Deep ensemble learning appears to one of the solutions that combine the predictions from multiple neural network models to reduce the variance of predictions and reduce generalization error. Recently, there are some studies to integrate base learners of deep neural networks with ensemble learning in three major ways. The first one is ensemble training data including re-sampling [25], bootstrap aggregation [26], where the choice of data is varied for training different base models in the ensemble. The second one is to ensemble models where different base models are used in the ensemble, including different random initialization, a random selection of mini-batches, differences in hyper-parameters, etc [16]. The third way is varying combinations where one vary the choice of combining outcomes from ensemble members. The most famous method is a model averaging ensemble and weighted average ensemble. Different from the aforementioned deep ensemble learning methods, deep *N*-ary ECOC serves a complementary piece for existing methods.

2.3 Deep Neural Networks

In recent years, a lot of different deep neural networks are proposed for different applications. For computer vision tasks, the most dominating model comes from Convolutional Neural Networks (CNNs), and its follow-up works such as AlexNet [15], VGGs [27], ResNet [28] and DenseNet [29]. For natural language processing tasks, most popular networks belong to Recurrent Neural Network (RNNs), or its many variants such as Long Short-Term Memory (LSTM) [30], Gated Recurrent Unit (GRU) [31] and etc. In the experiment, we validate deep *N*-ary ECOC in both CNNs and LSTMs architecture for vision and text datasets respectively.

3 Deep *N*-ary ECOC

In this section, we first introduce the concept of *N*-ary ECOCs. To facilitate the training procedure, we further propose three different parameter sharing architectures, namely full, partial and no sharing.

3.1 *N*-ary Ensemble for Multi-class Classification

Error correcting output codes (ECOC) constructs an ensemble of binary base classifiers by randomly and independently assigning positive/negative pseudo labels (i.e., $1/-1$ in the coding matrix) for each base task. The results of all the base learners are combined to make a prediction. ECOC consists of two main steps: 1) encoding 2) decoding. In encoding, we create an encoding matrix to encode each class into a unique code that is as different as possible from the codes of the remaining classes. One example of the encoding matrix is illustrated in Table 1(a). A row of the coding matrix represents the code of each class, while a column of the coding matrix represents the binary classes to be considered when learning a base classifier. In decoding, ECOC first computes the prediction vector that is the concatenation of results of all the base tasks. The final label is determined by assigning the label with the "closest" label vector in encoding matrix $M \in \{1, 2, \dots, N\}^{N_C \times N_L}$, where N_C denotes the number of classes and N_L denotes the number of base learners. As proved in many research works [12], [14], the capability of error correction relies on the minimum distance, $\Delta_{min}(M)$, between any distinct

pair of rows in the coding matrix M. In this way, the trained base classifiers could be sufficiently differentiated from each other.

To achieve this goal, ECOC is extended to a new framework of *N*-ary ECOC [14] where the original classes are decomposed into *N* meta-class ($3 \leq N \leq N_C$). Table 1(b) shows an example of *N*-ary ECOC encoding matrix. Zhou *et al.* [14] both empirically and theoretically showed that *N*-ary ECOC is able to achieve larger row separation and lower column correlation. It is interesting to note that *N*-ary ECOC is a more general framework for ECOC since traditional coding schemes could be treated as special cases of *N*-ary ECOC. For example, when $N = 2$, *N*-ary ECOC corresponds to the binary coding scheme; when $N = 3$, *N*-ary ECOC corresponds to the ternary coding scheme. Furthermore, recent works [16] showed that an Ensemble of models with different Random Initialization (ERI) is able to improve multi-class classification performance. This deep ensemble learning strategy could be also viewed as a special case in the framework of *N*-ary ECOC if we keep the original label assignment, namely $N = N_C$.

Table 1. Example of ECOC and *N*-ary coding matrix.

(a) ECOC

	M_1	M_2	M_3	M_4	M_5	M_6
C_1	-1	1	-1	1	1	1
C_2	1	1	-1	1	1	1
C_3	1	1	1	1	-1	-1
C_4	-1	1	1	1	-1	1
C_5	1	-1	1	1	-1	-1
C_6	1	1	1	-1	-1	1
C_7	-1	1	-1	-1	1	-1
C_8	1	1	-1	-1	1	1
C_9	1	1	1	-1	1	-1

(b) *N*-ary ECOC with $N = 4$

	M_1	M_2	M_3	M_4	M_5	M_6
C_1	1	1	2	4	1	1
C_2	2	1	1	3	2	1
C_3	3	2	1	2	3	1
C_4	4	3	1	1	4	2
C_5	4	3	2	2	4	3
C_6	4	3	3	3	3	4
C_7	3	4	4	4	2	4
C_8	2	4	3	4	2	3
C_9	3	4	2	3	3	2

On the other side, most existing work on ECOCs including our previous work on *N*-ary ECOC is constrained to classifier training with pre-defined features. With a significant advance of deep learning, the performance of various machine learning tasks has been improved. There is few works to discuss how to extend ECOC in the scenario of deep learning. In this work, we specifically study this open problem in the framework of *N*-ary ECOC, termed *Deep N-ary ECOC*, and propose several approaches to address it. In this paper, we mainly investigate the following three questions:

1. Do we necessarily independently train all the deep base learners from scratch for all the situation?
2. Whether the *N*-ary ECOC framework still retains the advantages over other data-independent ensemble approaches with deep neural network?
3. Any new suggestion on the choice of the meta-class number N and base learners number N_L?

For the first question, we are going to propose three different parameter sharing architectures, which is described in more details next section. For the remaining two questions, we delay the investigation in the experiment section.

3.2 Efficient Implementation for Deep *N*-ary ECOCs

Different from traditional ECOC problems with pre-defined features, Deep ECOCs should consider deep feature learning as well as the classifier construction during training. This increases the difficulty of deploying ECOCs in real-world scenarios, since even training a single deep neural network is also expensive. Fortunately, thanks to the nature of ECOCs, all the base deep neural networks could be trained simultaneously. Furthermore, in this paper, we investigate a more efficient realization and propose three different parameter sharing strategies, namely, no share, partial share, and full share, which is depicted in Figure 1.

Typically, we take the model of the CIFAR dataset, termed *CIFAR-CNN*s (explain in detail later), as an example to illustrate the three strategies. For the no parameter sharing strategy, as shown in Figure 1(a), we trained N_L base learners independently, which means that the feature encode layers of each base learner are trained by the inputs directly and do not interact with other learners. The partial parameter sharing strategy contains shared and task-specific layers, as in Figure 1(b), the first three feature encoder layers are shared by all the base learners while the top encoder layer is task-specific, which is only optimized by the corresponding meta-class objectives. The full parameter sharing strategy is simply set all the feature encode layers to be shared by all the base learners except the top classifiers (see Figure 1(c)). The top layer classifiers of all the sharing strategies are trained independently with its meta-class objectives. Note that, all the base learners of no parameters sharing strategy are trained from scratch while the shared layers of partial and full parameters sharing strategies are initialized by a pre-trained single model and fine-tuned through training to accelerate the model convergence rate. Obviously, the no parameter sharing strategy contains most parameters (N_n), then the partial sharing strategy (N_p) and the full sharing strategy (N_f) is least, say, $N_n > N_p > N_f$.

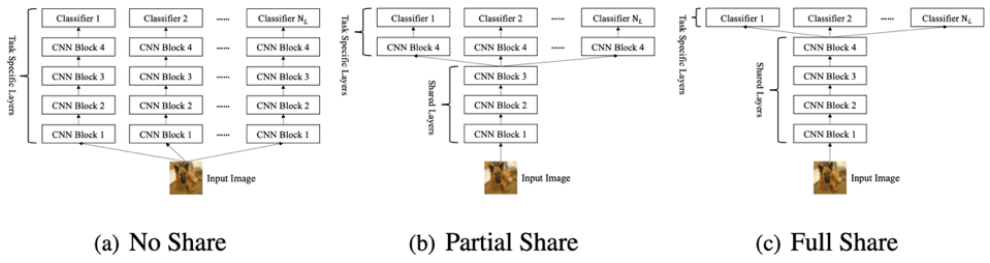


Figure 1. An example of three different parameters sharing strategies.

4 Experiments

4.1 Datasets

We conduct the experiments on 4 image datasets and 2 text datasets. The image datasets contain MNIST [32], CIFAR-10 [33], CIFAR-100 [33], and FLOWER-102 [34], which are

widely used image classification datasets in the computer vision community. While the text datasets are Text REtrieval Conference (TREC) [35] dataset and Stanford Sentiment Treebank (SST) [36] dataset. The TREC is the question and answering dataset which involves classifying question sentences into 6 question types, say, whether the question is about person, location, numeric information and etc. The SST is the sentiment analysis sentence data with 5 classes that range from 0 (most negative) to 5 (most positive). The statistics of these datasets are described in Table 2. Note that we do not utilize the *K*-fold cross-validation method, but simply use the split of train/validation/test sets. If the datasets do not contain development part, we randomly split 10% training samples as the development dataset.

Table 2. Statistics of Image and Text Datasets.

Image Dataset					
Dataset	Image Size	# Train Sample	# Dev Sample	# Test Sample	# Classes (N_C)
MNIST	28×28	60,000	N/A	10,000	10
CIFAR-10	32×32	50,000	N/A	10,000	10
CIFAR-100	32×32	50,000	N/A	10,000	100
FLOWER-102	256×256	6,552	818	819	102
Text Dataset					
Dataset	Avg. Sent. Len.	# Train Sample	# Dev Sample	# Test Sample	# Classes (N_C)
TREC	10	5,500	N/A	500	6
SST	18	11,855	N/A	2,210	5

4.2 Experimental Setup

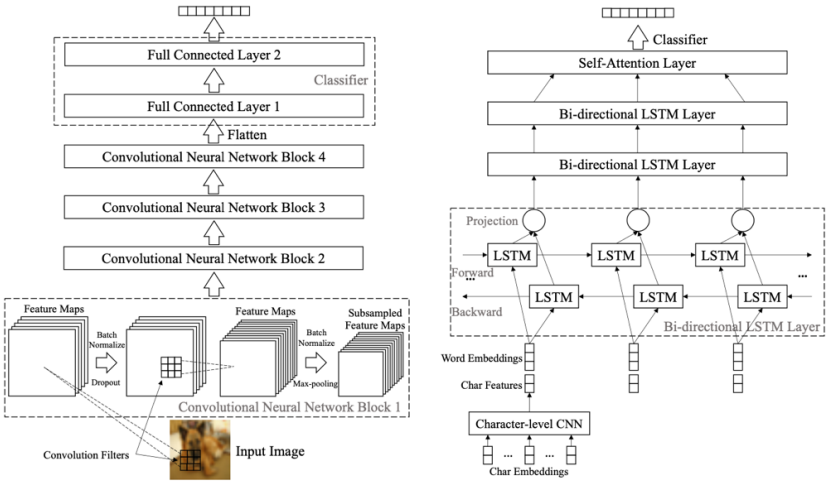
Deep Neural Networks. We employ different neural network-based models for different datasets. Specifically, we use LeNet [32] for the MNIST dataset and the FLOWER-102 dataset is trained by AlexNet [15]. Note that, due to the difficulty for the AlexNet model to learn consequential and representative features from the small training dataset of FLOWER-102 directly, the AlexNet is not trained from scratch but obtained by fine-tuning the pre-trained AlexNet model[3], which is trained on ILSVRC dataset. For CIFAR-10/100 datasets, we build a model with eight convolutional layers and two full-connected layers, named as *CIFAR-CNNs*, as shown in Figure 2(a), where the eight convolutional layers are divided into four groups, they share the same structure with the different numbers of filters and kernel widths. The architecture of each group is structured as follows: one convolutional layer following the batch normalization [37] and dropout [38] layer, another convolutional layer with batch normalization and max-pooling is applied. And the ELU [39] activation function is used for each convolutional layer.

To train the TREC and SST text datasets, we construct a three-layer bidirectional LSTM model with character-level CNN [40] and self-attention [41] mechanism, termed *Bi-LSTMs*, as shown in Figure 2(b), where the character-level CNN learned the character features to represent a word from the character sequences of such word, which can help to enrich the meaning of word features, especially for rare and out-of-vocabulary words, and boost the performance by capturing morphological and semantic information, and the self-attention mechanism encodes the learned contextual affluent word-level feature sequence of

[3] Pre-trained AlexNet: http://www.cs.toronto.edu/~guerzhoy/tf_alexnet/

bidirectional LSTM into a single vector by considering the importance of each word-level feature.

Parameters Setup. For LeNet of MNIST dataset, we follow the same settings as LeCun *et al.* [32] and RMSProp [42] is chosen as the parameters optimization method with a learning rate of 0.001 and decay rate of 0.9, we also introduce Dropout [38] strategy with a drop rate of 0.5 at each convolutional layer and the first full-connected layer to prevent over-fitting. While the AlexNet for FLOWER-102 dataset, we utilize exactly the same structure, parameter setting, and optimization method as Alex *et al.* [15]. For *CIFAR-CNNs* model of CIFAR-10/100 datasets, we set the number of filters for each convolutional block as 32, 64, 128, and 256, respectively, the kernel sizes of $(3, 3)$ and pool size of $(2, 2)$ for all the blocks. The hidden size of the first fully-connected layer is 512, while the second depends on the class size. To avoid over-fitting, we apply ℓ_2 regularization with weight decay rate of 0.0005 for all the weight parameters and Dropout strategy, the drop rate is 0.3, 0.4, 0.4, 0.4 for each convolutional block respectively, and 0.5 for the first fully-connected layer. Parameters optimization is performed by Adam optimizer [43] with gradient clipping of 5.0 and learning rate decay strategy. We set the initial learning rate of $\beta_0 = 0.002$ and fixed it for the first 5000 training iterations, then the learning rate β_i is updated by $\beta_i = \beta_0/(1 + \rho \times \frac{t-5000}{T})$, where T is the decay step of 500 and ρ is the decay rate of 0.05. Meanwhile, in order to improve the performance, the data augmentation is also utilized.



(a) CIFAR-CNNs model (b) Bi-LSTMs model

Figure 2. The general architecture of *CIFAR-CNNs* and *Bi-LSTMs* models.

For the *Bi-LSTMs* model of TREC and SST text datasets, we use the 300-dimensional publicly available pre-trained word embeddings as the word-level feature representation, which is trained by fastText[4] package on *Common Crawl* and *Wikipedia* [44], [45], and the 50-dimensional randomly initialized task-specific character embeddings. The word embeddings are fixed and character embeddings are learned during training. We use three different

[4] fastText: https://github.com/facebookresearch/fastText

convolutional layers with widths $2, 3, 4$, respectively, for character-level CNN encoder and set the filter number of each layer as 20, the learned character features of each layer are concatenated and then optimized by a two-layer highway network [46] before concatenating with the corresponding word embeddings. The dimension of hidden states of LSTM layers are set as 200. Parameters optimization is performed by Adam optimizer [43] with gradient clipping of 5.0 and learning rate decay strategy. We set the initial learning rate of $\beta_0 = 0.001$, at each epoch t, learning rate β_t is updated by $\beta_t = \beta_0/(1 + \rho \times t)$, where ρ is the decay rate with 0.05. To reduce overfitting, we also apply Dropout [38] at the embedding layer and the output of each LSTM layer with the drop rate of 0.2 and 0.3, respectively.

***N*-ary ECOC Coding Matrix Setup.** For *N*-ary ECOCs, including ECOCs (which is a special issue of *N*-ary ECOC when $N = 2$), we train the N_L base learners based on the coding matrix and use the predicted code sequence of each class and generated coding matrix to make a prediction based on distance measurement. Zhou *et al.* [14] introduced several coding matrix construction methods and distance measurements designed for general or task-specific applications. For simplicity, we utilize the random dense encoding method to randomly split the original classes N_c into N subsets and make sure that the number of classes in each subset should be approximately balanced, simultaneously. For the decoding method, we adopt the minimum Hamming distance due to its simplicity and effectiveness. In our experiments, we experiment on the different numbers of meta-class N and number of base learners N_L for different datasets, as described in Table 3. Note that we do not experiment on all the possible meta-classes for each dataset, because of the limitations of computing resources and we only trained 60 base learners for MNIST, FLOWER-102, TREC, and SST datasets and 100 base learners for CIFAR-10/100 datasets, respectively. Specifically, in order to evaluate the effects of number of base learners on the ensemble learning performance, we trained another 300 classifiers for FLOWER-102 and TREC datasets, respectively.

Table 3. Summarization of tested N and N_L for experiments.

Dataset	# Classes (N_C)	Tested # Meta-Class (N)	Tested # Base Learners* (N_L)
MNIST	10	$2, 4, 5, 8, 10$	60
CIFAR-10	10	$2, 4, 5, 8, 10$	100
CIFAR-100	100	$2, 5, 10, 30, 50, 75, 95, 100$	100
FLOWER-102	102	$2, 3, 5, 10, 20, 40, 60, 80, 90, 95, 102$	60
TREC	6	$2, 3, 4, 5, 6$	60
SST	5	$2, 3, 4, 5$	60

* It indicates the maximal number of classifiers is used for training.

4.3 Experimental Results

4.3.1 Comparison with Different Ensemble Methods

In this section, we compare the performance of different ensemble methods on the aforementioned image and text datasets. In the experiment, we trained a single model and ensemble models of three coding schemes for each dataset, i.e., Ensemble with Random Initializations (ERI), ECOC, *N*-ary ECOC, then report their (ensemble) accuracy with standard deviations. Note that we only report the highest score under a specific meta-class N for *N*-ary ECOC. For the MNIST, FLOWER-102, TREC, and SST datasets, we use 60 base

learners for each scheme, while 100 base learners for CIFAR-10/100 datasets. The results are summarized in Table 4. Generally, we observe that most ensemble models show relatively significant improvements, compared with the single model, on the given datasets with different deep neural networks.

We observe two interesting results in Table 4. First, comparing the single model with *N*-ary ECOC, we find that the improvement ratio of *N*-ary ECOC is inverse relation with single model performance, i.e., the improvement of *N*-ary ECOC scheme is more prominent if the performance of the single model is lower. For example, it is obvious that the baseline accuracies are higher on MNIST, CIFAR-10, TREC and FLOWER-102 ($> 80\%$) than on CIFAR-100 and SST (almost $< 60\%$), then the improvement ratios are 0.59%, 5.54%, 5.64% and 5.80% from the single model to *N*-ary ECOC on MNIST, CIFAR-10, TREC, and FLOWER-102 datasets, respectively, while the improvement ratios of CIFAR-100 dataset are 13.72% and 15.15% on SST dataset.

Table 4. Ensemble accuracies with their standard deviations.

Dataset	Method	Single Model	Ensemble Model*		
			ERI	ECOC	*N*-ary ECOC
MNIST	LeNet [32]	98.98 ± 0.07%	99.11 ± 0.11%	99.23 ± 0.08%	**99.57** ± 0.09%
CIFAR-10	CIFAR-CNNs	87.12 ± 0.43%	90.54 ± 0.31%	89.37 ± 0.54%	**91.95** ± 0.24%
CIFAR-100	CIFAR-CNNs	61.50 ± 0.57%	69.57 ± 0.29%	34.26 ± 2.42%	**69.94** ± 0.32%
FLOWER-102	AlexNet [15]	83.12 ± 0.29%	86.32 ± 0.60%	77.05 ± 0.73%	**87.94** ± 0.28%
TREC	Bi-LSTMs	90.50 ± 0.12%	94.80 ± 0.09%	95.80 ± 0.08%	**95.60** ± 0.10%
SST	Bi-LSTMs	44.17 ± 0.92%	48.69 ± 0.18%	48.91 ± 0.26%	**50.86** ± 0.13%

* Here N_L are 60, 100, 100, 60, 60 and 60, respectively, for the ensemble models from top to bottom row. While *N* are 3, 4, 95, 95, 3, 4, respectively, for the *N*-ary ECOC.

Second, the *N*-ary ECOC scheme outperforms ECOC and ERI ensemble methods on most image and text datasets, except for the TREC text dataset. Specifically, *N*-ary ECOC always performs better than ERI. This is due to that *N*-ary ECOC varies the predicted classes for each base learner and makes them more diverse than ERI, where the diverse forecast errors made by base learners of *N*-ary ECOC are more beneficial to the ensemble learning in comparison to the similar base learner errors of ERI. Meanwhile, compared with ECOC, *N*-ary ECOC also shows its superiority in most cases, especially when the number of classes is large (i.e., $N_c \geq 100$ in our experiments). It is primarily due to, as mentioned by Zhou *et al.* [14], the better quality of the coding matrix and the higher discriminative ability (in terms of how many meta-classes a base learner tries to discriminative) of *N*-ary ECOC than ECOC.

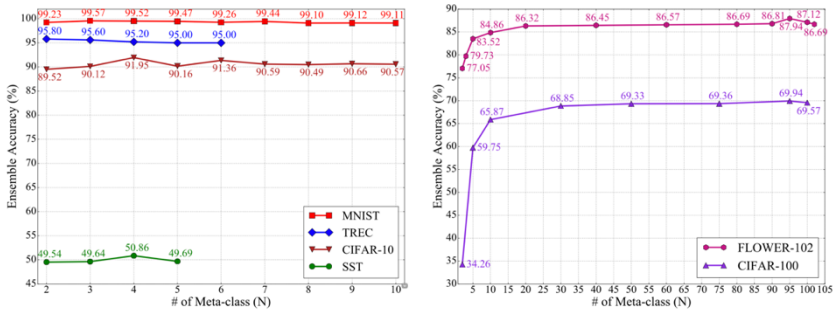
In fact, we find the contribution of class merge degree to the ensemble accuracy of *N*-ary ECOC replies on the dataset, say, the datasets with a different number of classes require different class merge degree strategy, as discuss in Section 4.3.2. Note the class merge degree, which is measured by $\frac{N_c - N}{N_c}$, is the ratio of class numbers reduced when the classes are merged into meta-classes.

4.3.2 Evaluation on the Effect of Meta-class Number *N*

In this section, we investigate the influence of meta-class number N, which is one of the crucial hyper-parameters of *N*-ary ECOC. For the datasets with a small value of N_c, we experiment on all the possible meta-class numbers, i.e., from 2 to N_c ($N = 2$ denotes ECOC

and $N = N_c$ denotes ERI), while for the datasets with a large value of N_c, we select several representative meta-class numbers for the experiment. The ensemble accuracies with respect to N are depicted in Figure 3.

From Figure 3(a), we observe that the performances of ensemble models with different N are relatively stable, the highest ensemble accuracies of MNIST, CIFAR-10, and SST achieve when $N = 3$, $N = 4$, and $N = 4$ respectively, and the best performance of TREC is obtained at $N = 3$ if we do not consider the ECOC. After that, the performance of each dataset is gradually decreased with small fluctuations with an increasing number of meta-class N. It is interesting to see that N-ary ECOC for datasets with a small value of N_c always tends to arrive the best performance with small value of N, i.e., large class merge degree. Specifically, the class merge degree for MNIST, CIFAR-10, TREC and SST are 0.7, 0.6, 0.5 and 0.2 respectively.



(a) Datasets with small value of N_c (b) Datasets with large value of N_c

Figure 3. Ensemble accuracies with respect to N, where the first point of each line represents ECOC ($N = 2$), the last represents ERI ($N = N_c$) and the rest is N-ary ECOC with various N.

However, as shown in Figure 3(b), the performance of ensemble models with different N fluctuates significantly on the datasets with a large value of N_c. For the ECOC scheme, it only achieves 34.26% ensemble accuracy on the CIFAR-100 dataset and 77.05% on the FLOWER-102 dataset. For the N-ary ECOC with $N = 3$, it obtains 83.52% and 59.75% on FLOWER-102 and CIFAR-100 datasets, respectively. Then the ensemble accuracy improves gradually with the increase of meta-class N and reaches the summit with accuracies of 87.94% and 69.94%, when $N = 95$, for FLOWER-102 and CIFAR-100 datasets, and mildly decreases after the optimal performances. The ensemble accuracies of the ERI scheme ($N = N_c$) on these two datasets are slightly lower than that of N-ary ECOC. Obviously, the ECOC fails to address the datasets with a large value of N_c, while the higher ensemble performance of N-ary ECOC needs a large value of N, namely, a small class merge degree. For the FLOWER-102 and CIFAR-100 datasets, the N-ary obtains good results after $N \geq 75$ and achieve best at $N = 95$. In particular, the class merge degree for FLOWER-102 is 0.069 and CIFAR-100 is 0.05.

In general, we conclude from the experiment that the ensemble performance is relatively stable for the datasets with a small value of N_c, which slightly improves until the peak and then decreases a bit, or just slightly decreases with the increase of N after the peak. While the performance, for the datasets with a large value of N_c, boosts significantly at the very beginning, then it saturates as N continues increasing and reaching the optimum when N is

close to N_c. This could be explained by that the base learners with large N has stronger discriminability [14].

Thus, our suggestions for the choice of N are: 1) For the dataset with small N_c, the large class merge degree strategy, i.e. small N, is better for achieving good performance, such as $N = 3$ or 4 for the dataset with $N_c \leq 10$. 2) Reversely, for the dataset with large N_c, the small class merge degree strategy should be applied, e.g. $75 \leq N \leq 95$ for N_c is around 100.

4.3.3 Evaluation on the Effect of Base Learner Number N_L

In this experiment, we further explore another crucial hyper-parameter of *N*-ary ECOC, namely the number of base learner N_L (also equivalent to the code length), and study its influence on the ensemble accuracy. We first report the ensemble accuracies of different N_L for each dataset with the optimal meta-class number N, as described in Table 5. Then, we study the ensemble accuracies of different meta-class N with respect to N_L (see Figure 4 and 5).

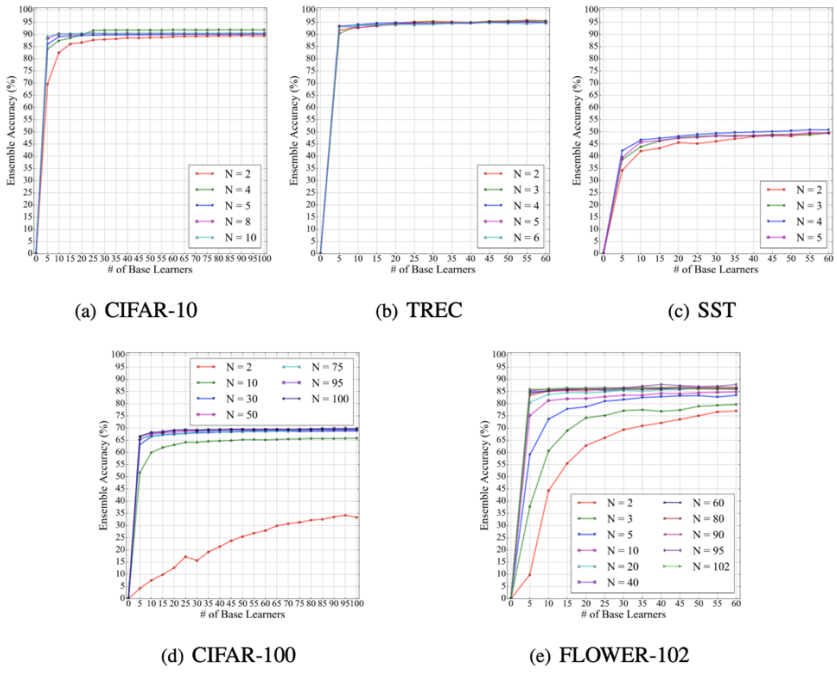
From Table 5, we observe that one requires a smaller number of base learners N_L for datasets with small N_c than that for datasets with large N_c to reach the optimal ensemble accuracies generally. For example, MNIST and TREC only need 50 base learners to get the optimum, while SST obtains best accuracies with 60 base learners and CIFAR-10 requires 80. In comparison, it reaches the optimal ensemble accuracies with the help of 100 base learners on CIFAR-100 (note that it first reaches optimum when $N_L = 90$). There is a special issue that FLOWER-102 holds a large N_c (102 classes), but only requires 60 base learners to derive the optimal ensemble accuracies. It is because the pre-trained model on the large-scale dataset (ILSVRC dataset in our experiment) is utilized and the pre-trained model already encodes a variety of abstractly and typically well-learned features. Moreover, we also find that the requirement of N_L is related to the single model performance to some degree, say, the single model achieves better performance, then its ensemble model requires fewer base learners to achieve the optimal result.

Table 5. Ensemble accuracies of *N*-ary ECOC with respect to number of base learners N_L.

Dataset	N	# of Base Learners (N_L)							
		10	20	30	45	50	60	80	100
MNIST	3	99.14%	99.20%	99.35%	99.48%	99.57%	99.57%	-	-
CIFAR-10	4	87.45%	89.76%	91.78%	91.83%	91.82%	91.92%	**91.95%**	91.93%
CIFAR-100	95	67.94%	69.12%	69.11%	69.33%	69.34%	69.46%	69.67%	**69.94%**
FLOWER-102	95	86.06%	86.45%	86.45%	87.06%	87.16%	**87.94%**	87.46%	87.59%
TREC	3	93.80%	94.00%	95.20%	95.20%	95.60%	**95.60%**	95.50%	**95.60%**
SST	4	46.74%	48.19%	49.41%	50.18%	50.45%	**50.86%**	-	-

In addition to the observations from Table 5, we also study the impact of N_L on ensemble performance under different meta-class N. Obviously, the optimal number of base learners N_L for achieving the best accuracy is related to the meta-class N, as shown in Figure 4 and 5. Normally, an ensemble model with small meta-class N needs more base learners N_L to achieve the same result compared to an ensemble model with large meta-class. It is because the discriminative ability of the codes for small N is worse than that for large N.

Considering the definition of ECOC, *N*-ary ECOC and ERI schemes, the discriminative ability of ECOC is worst due to its small meta-class ($N = 2$), which means ECOC needs relatively most base learners to reach the optimal performance compared with *N*-ary ECOC and ERI, where ERI holds the best discriminative ability. Thus, we conclude that N_L for ECOC is greater than or equal to N_L for *N*-ary ECOC, while N_L for *N*-ary ECOC is greater than or equal to N_L for ERI. Note that we use "*greater than or equal to*" since there is no guarantee that the optimal N_L for a small N must be larger than that for a large N, especially for some extreme situations such as $N = 2$ (ECOC) versus $N = 3$ (*N*-ary ECOC) or $N = 99$ (*N*-ary ECOC) versus $N = 100$ (ERI) for the dataset with 100 classes (N_c).



(a) CIFAR-10 (b) TREC (c) SST

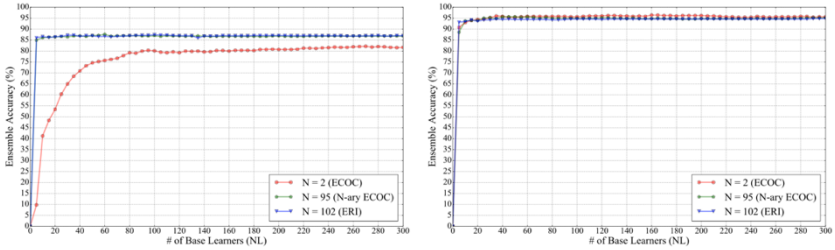
(d) CIFAR-100 (e) FLOWER-102

Figure 4. Ensemble accuracies of different values for N with respect to N_L on the image and text datasets, where $N = 2$ is ECOC scheme, $N = N_c$ (biggest N in each sub-figure) is ERI scheme, and the rest is *N*-ary ECOC schemes.

In Figure 4, the experiment results on all the datasets show similar trends that ensemble accuracies of larger N converge faster than that of smaller N as the increasing of N_L, which means larger N requires less N_L and vice versa. For example, as shown in Figure 4(a), ECOC reaches optimal ensemble accuracies at $N_L = 100$, while *N*-ary ECOC with $N = 4$ and 5 optima at $N_L = 80$, then $N_L = 60$ for $N = 8$ and ERI peaks at $N_L = 55$. The patterns on TREC and SST datasets are consistent as CIFAR-10. Typically, such patterns are more distinct for datasets with large N_c (ref. Figure 4(d) and 4(e)). For instance, in Figure 4(d), the ensemble accuracies of $N = 10$ are highest at $N_L = 100$, $N_L = 95$ for *N*-ary ECOC with $N = 30,50,75$, while 90 base learners are required for $N = 95$ and ERI needs $N_L = 85$. Here we do not take ECOC into consideration, since it fails to improve the ensemble accuracy with only 34.26%. For the Figure 4(e), we see that ensemble accuracies of $N = 2, 3, 5, 10$

converge at $N_L = 60$, $N = 20, 40$, converges at around $N_L = 50$, $N = 60, 80, 90$ at approximately $N_L = 45$ and 40 base learners are needed for *N*-ary ECOC with $N = 95$ and ERI to reach the optimal ensemble accuracy.

Apart from the optimal N_L for each meta-class N to reach optimal ensemble accuracy, we also observe that using $15\sim25$ base learners for ERI is good enough for datasets with small N_c while $20\sim40$ base learners for large N_c. For ECOC, it fails with the large N_c, and on the dataset with small N_c. Although ECOC performs comparably to *N*-ary ECOC and ERI, it still needs more base learners to converge, which is different from the conclusion in [13] that ECOC requires $N_L = 10\log_2(N_c)$ on traditional classifiers. For *N*-ary ECOC, the optimal performance is highly related to the choice of N. If the choice of N follows suggestions in Section 4.3.2, $40\sim60$ base learners for small N_c are enough to achieve good performance, while large N_c needs around $60\sim100$ base learners.



(a) FLOWER-102 (b) TREC

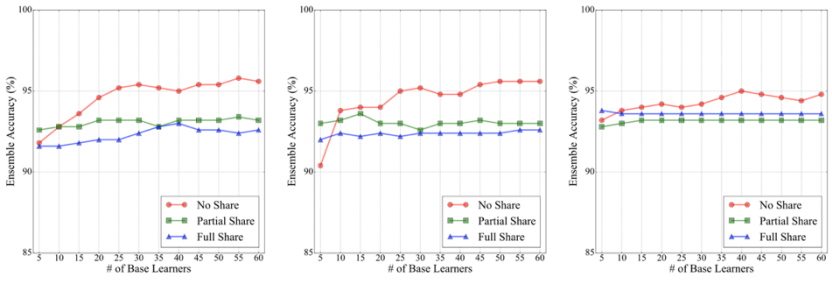
Figure 5. Ensemble accuracies with respect to large $N_L(= 300)$ for three coding schemes on FLOWER-102 and TREC datasets.

We further extend number of base learners to 300 and experiment on the FLOWER-102 and TREC datasets to investigate ensemble performances with the increasing N_L, as in Figure 5. From Figure 5(a), we find that the performance of ECOC improves significantly when N_L increases, then keep relatively stable with a slight increase after $N_L = 100$ and reach the optimal accuracy of 82.17% at around $N_L = 270$. However, the best performance of ECOC derived by using a large number of base learners is still lower than *N*-ary ECOC with $N = 95$ and ERI with only 5 base learners used, which indicates that ECOC is not suitable for the large N_c case. For the *N*-ary ECOC and ERI, they obtain good scores with only small numbers of base learners and slightly improve to the optimal accuracy at around $N_L = 40$. After that, the performance remains stable with the increase of N_L and it drops when N_L continues to increase, which indicates that increasing N_L monotonously has no impact on performance. Similar observations could be found in Figure 5(b).

Generally, there is no concrete conclusion for the choice of the number of base learners N_L, but some helpful guidelines can be summarized for experiments: 1) The choice of meta-class N is more important than the number of base learners N_L for the performance of *N*-ary ECOC, especially for the dataset with large N_c. Since the increase of N_L cannot compensate for the negative effects caused by a badly selected N (e.g., $N = 10$ for CIFAR-100). 2) Albeit the optimal number of base learners N_L varies along N_c, the suggested N_L is in the range of $[\lfloor 10\log_{2.2}(N_c)\rfloor, \lceil 10\log_{1.5}(N_c)\rceil]$. For example, the optimal N_L ranges in $[30, 58]$ for $N_c = 10$ and $[59, 110]$ for $N_c = 100$, which aligns with the observations in our experiments.

4.3.4 Comparison with Three Parameter Sharing Strategies

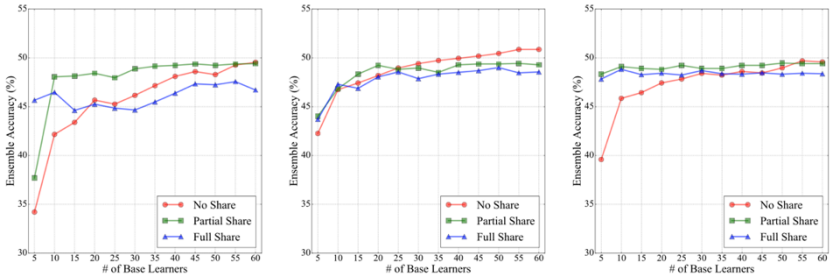
In this Section, we study the effect of three different parameter sharing strategies in the framework of ECOC, *N*-ary ECOC, and ERI. Note that, for the *N*-ary ECOC framework, we only select the optimal meta-class N of each dataset for display except for the CIFAR-100 dataset which four different N are chosen for display. We first study the performance of three different parameter sharing strategies on each tested dataset.



(a) ECOC ($N = 2$) (b) N-ary ECOC ($N = 3$) (c) ERI ($N = 6$)

Figure 6. Parameter sharing strategies in ECOC, *N*-ary ECOC and ERI for TREC dataset.

From the experimental results on the TREC dataset (see Figure 6), we observe that no parameter sharing strategy performs better than partial and full parameter sharing strategy for ECOC, *N*-ary ECOC, and ERI. When the number of base learner N_L is small, the performance of no share is not satisfactory. Then it improves significantly with the increase of N_L, while the performances of partial and full share are relatively stable with respect to N_L. Moreover, when the number of meta-class N is small, partial share outperforms the full share and the performance of no share is much better than partial and full share. However, when N is large, full share is better than partial share and the performance of no share is just slightly higher than partial and full share.

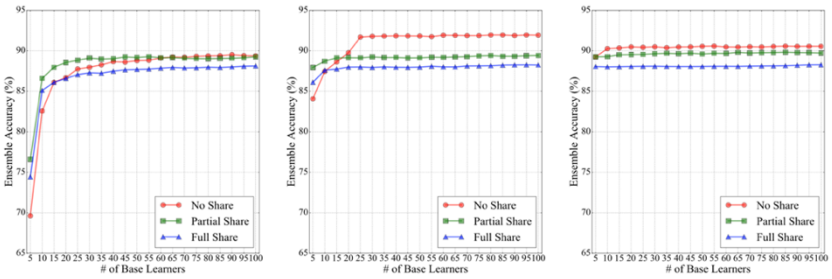


(a) ECOC ($N = 2$) (b) N-ary ECOC ($N = 4$) (c) ERI ($N = 5$)

Figure 7. Parameter sharing strategies in ECOC, *N*-ary ECOC and ERI for SST dataset.

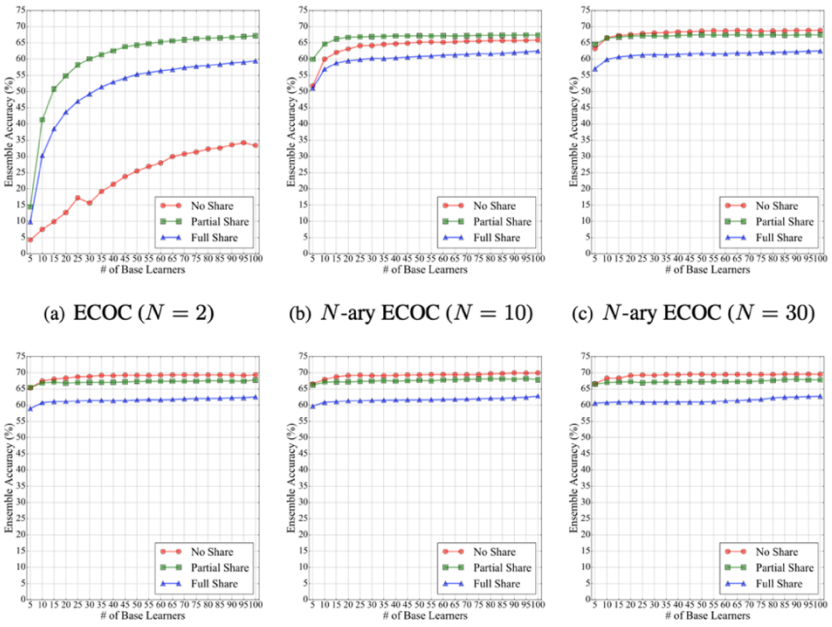
From Figure 7, we have the following observations. First, when the number of meta-class N is small, both partial and no share models improve significantly with the increase of N_L. The partial share generally outperforms the no and full share except when N_L is less. Second, when the number of meta-class N is large, as shown in Figure 7(b) and 7(c), the performance of the

three strategies are stable, and the improvement of no share is most significant with the increase of N_L. No share strategy governs the best performance with $N = 4$ while partial share strategy always performs best for ERI situation.



(a) ECOC ($N = 2$) (b) N-ary ECOC ($N = 4$) (c) ERI ($N = 10$)

Figure 8. Parameter sharing strategies in ECOC, *N*-ary ECOC and ERI for CIFAR-10 dataset.



(a) ECOC ($N = 2$) (b) N-ary ECOC ($N = 10$) (c) N-ary ECOC ($N = 30$)

(d) N-ary ECOC ($N = 50$) (e) N-ary ECOC ($N = 95$) (f) ERI ($N = 100$)

Figure 9. Parameter sharing strategies in ECOC, *N*-ary ECOC and ERI for CIFAR-100 dataset.

In Figure 8, the performances of no, partial, and full share strategies are more stable. When the number of base learners N_L is small, we see that the performance of no share is worst with ECOC and *N*-ary ECOC, and partial share performs better with *N*-ary ECOC and ERI situations. With the increase of N_L, for ECOC, all the strategies improve significantly, partial share outperforms another two strategies at the beginning, and then no share comes closer to partial share and reaches slightly higher performance than partial share. For *N*-ary ECOC, partial and full share strategies do not show significant improvement, while no share

improves obviously and outperforms the partially and full share despite its lower ensemble accuracy at the very beginning. For the ERI, all these three strategies perform stable while no share always performs best and the performance of full share stays the bottom.

In the last experiment, we study the parameter sharing strategies in ECOC, *N*-ary ECOC, and ERI for the dataset with a large number of classes, as shown in Figure 9. For *N*-ary ECOC situation, we experiment on four different meta-class with $N = 10, 30, 50, 95$.

First, we observe that ECOC model with no share strategy fails to achieve satisfactory performance, while partial and full share strategies with the ECOC improve significantly with the increase of N_L. Moreover, partial share always outperforms full share.

Secondly, for the *N*-ary ECOC with small number of meta-class, we observe that partial share strategy outperforms no and full share always. No share improves most significantly and its performance is comparable to that of partial share with the increase of N_L. The performance of full share always maintains the worst. With an increasing number of meta-class *N*, partial share strategy outperforms no share strategy at the beginning, but its performance is gradually surpassed by no share when number of base learners N_L increases. For $N = 50$ and 95, the performance of no share is comparable to that of partial share when the number of base learners N_L is small. No share outperforms partial share with the increases of N_L. Moreover, for the N-ary ECOC, full share strategy consistently performs worst.

Thirdly, for the ERI model, the observations are similar to the *N*-ary ECOC with large meta-class *N* and the no share strategy is comparable to partial share when N_L is small. It always performs best when N_L increases, meanwhile, the performance of full share is worst.

Finally, we conclude that: 1) In general, for the dataset with the small number of classes, the performance of no share model is better than or equal to that of the partial share model, thus no share strategy is suggested to be chosen. 2) For the dataset with the small number of classes, when the number of meta-class N is large, these three strategies perform stable. 3) For the dataset with a large amount of classes, when the number of meta-class is small, the performance of partial share model is the best. 4) For the dataset with large amount of classes, when the number of meta-class is large, no share strategy model outperforms partial and full share models in most cases. Thus no share strategy should be preferred in such a case. 5) If the number of meta-class is large, the performance difference between three sharing strategies is marginal. Then full share could be suggested due to its parameter efficiency.

5 Conclusion

In this paper, we mainly investigate how to effectively integrate deep learning with the *N*-ary ECOC framework, also termed Deep *N*-ary ECOC. To achieve this goal, we give three different realizations. We further carry out extensive experiments to show the superiority of deep *N*-ary ECOC over existing data-independent deep ensemble strategies.

Acknowledgement. The paper is supported by the Agency for Science, Technology and Research (A*STAR) under its AME Programmatic Funding Scheme (Project No. A18A1b0045). Ivor W. Tsang was supported by ARC DP180100106 and DP200101328.

References

[1] D. G. Lowe: Object recognition from local scale-invariant features. The proceedings of the seventh IEEE international conference on computer vision. Vol. 2, pp. 1150–1157 (1999).
[2] M. Riesenhuber, T. Poggio, Hierarchical models of object recognition in cortex. Nature neuroscience 2 (11) (1999) 1019.
[3] P. F. Felzenszwalb, D. P. Huttenlocher, Pictorial structures for object recognition, International journal of computer vision 61 (1) (2005) 55–79.
[4] J. Kittler, R. Ghaderi, T. Windeatt, J. Matas, Face verification via error correcting output codes., Image Vision Comput. 21 (13-14) (2003) 1163–1169.
[5] B. Pang, L. Lee, S. Vaithyanathan, Thumbs up?: Sentiment classification using machine learning techniques, in: Proceedings of the ACL-02 Conference on Empirical Methods in Natural Language Processing - Volume 10, EMNLP '02, Association for Computational Linguistics, 2002, pp. 79–86.
[6] X. Glorot, A. Bordes, Y. Bengio, Domain adaptation for large-scale sentiment classification: A deep learning approach, in: Proceedings of the 28th international conference on machine learning (ICML-11), 2011, pp. 513–520.
[7] J. Deng, S. Satheesh, A. Berg, L. Fei-Fei, Fast and balanced: Efficient label tree learning for large scale object recognition, in: NIPS, 2011.
[8] K. Torkkola, Linear discriminant analysis in document classification, in: In IEEE ICDM Workshop on Text Mining, 2001.
[9] Y. Freund, R. E. Schapire, Large margin classification using the perceptron algorithm, Machine Learning 37 (3) (1999) 277–296.
[10] C.-C. Chang, C.-J. Lin, Libsvm: A library for support vector machines, ACM Trans. Intell. Syst. Technol. 2 (3) (2011) 27:1–27:27.
[11] T. G. Dietterich, G. Bakiri, Error-correcting output codes: A general method for improving multiclass inductive learning programs, in: AAAI, AAAI Press, 1991, pp. 572–577.
[12] T. G. Dietterich, G. Bakiri, Solving multiclass learning problems via error-correcting output codes, J. Artif. Int. Res. 2 (1) (1995) 263–286.
[13] E.L.Allwein,R.E.Schapire,Y.Singer,Reducingmulticlasstobinary:aunifying approach for margin classifiers, J. Mach. Learn. Res. 1 (2001) 113–141.
[14] Zhou, J. T., Tsang, I. W., Ho, S. S., & Müller, K. R. (2019). N-ary decomposition for multi-class classification. Machine Learning, 108(5), 809-830.
[15] A. Krizhevsky, I. Sutskever, G. E. Hinton, Imagenet classification with deep convolutional neural networks, in: Advances in Neural Information Processing Systems 25, 2012, pp. 1097–1105.
[16] Y. B. Ian Goodfellow, A. Courville, Deep learning, book in preparation for MIT Press (2016). URL: http://www.deeplearningbook.org
[17] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, L. Fei-Fei, ImageNet: A Large-Scale Hierarchical Image Database, in: CVPR09, 2009.
[18] R. E. Schapire, The strength of weak learnability, Machine Learning 5 (2) (1990) 197–227.
[19] O. Pujol, P. Radeva, J. Vitria, Discriminant ECOC: a heuristic method for application dependent design of error correcting output codes, in: IEEE Transaction on Pattern Analysis and Machine Intelligence, 2006, pp. 1007–1012.
[20] S. Escalera, O. Pujol, P. Radeva, On the decoding process in ternary error-correcting output codes, Pattern Analysis and Machine Intelligence, IEEE Transactions on 32 (1) (2010) 120–134.
[21] M. A. Bagheri, G. A. Montazer, E. Kabir, A subspace approach to error correcting output codes, Pattern Recognition Letters 34 (2) (2013) 176 – 184.
[22] S. Escalera, O. Pujol, Ecoc-one: A novel coding and decoding strategy., in: ICPR, 2006, pp. 578–581.
[23] G. Zhong, M. Cheriet, Adaptive error-correcting output codes, in: IJCAI, 2013, pp. 1932–1938.
[24] Y. Freund, R. E. Schapire, A decision-theoretic generalization of on-line learning and an application to boosting, Journal of Computer and System Sciences 55 (1) (1997) 119–139.
[25] B. Efron, The jackknife, the bootstrap, and other resampling plans, Vol.38, Siam, 1982.
[26] L. Breiman, Bagging predictors, Machine learning 24 (2) (1996) 123–140.

[27] K. Simonyan, A. Zisserman, Very deep convolutional networks for large-scale image recognition, CoRR abs/1409.1556.
[28] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2016) 770–778.
[29] G. Huang, Z. Liu, L. van der Maaten, K. Q. Weinberger, Densely connected convolutional networks, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017.
[30] S. Hochreiter, J. Schmidhuber, Long short-term memory, Neural Comput. 9 (8) (1997) 1735–1780.
[31] K. Cho, B. van Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, Y. Bengio, Learning phrase representations using rnn encoder–decoder for statistical machine translation, in: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), Association for Computational Linguistics, 2014, pp. 1724–1734.
[32] Y. Lecun, L. Bottou, Y. Bengio, P. Haffner, Gradient-based learning applied to document recognition, Proceedings of the IEEE 86 (11) (1998) 2278–2324.
[33] A. Krizhevsky, G. Hinton, Learning multiple layers of features from tiny images, Master's thesis, Department of Computer Science, University of Toronto.
[34] M.-E. Nilsback, A. Zisserman, Automated flower classification over a large number of classes, in: Proceedings of the Indian Conference on Computer Vision, Graphics and Image Processing, 2008.
[35] X. Li, D. Roth, Learning question classifiers, in: COLING 2002: The 19th International Conference on Computational Linguistics, 2002.
[36] R. Socher, A. Perelygin, J. Wu, J. Chuang, C. D. Manning, A. Ng, C. Potts, Re- cursive deep models for semantic compositionality over a sentiment treebank, in: Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, Association for Computational Linguistics, 2013, pp. 1631–1642.
[37] S. Ioffe, C. Szegedy, Batch normalization: Accelerating deep network training by reducing internal covariate shift, CoRR abs/1502.03167.
[38] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, R. Salakhutdinov, Dropout: A simple way to prevent neural networks from overfitting, JMLR (2014) 1929–1958.
[39] D.-A. Clevert, T. Unterthiner, S. Hochreiter, Fast and accurate deep network learning by exponential linear units (elus), in: International Conference on Learning Representations, 2016.
[40] Y. Kim, Y. Jernite, D. Sontag, A. M. Rush, Character-aware neural language models, in: Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, 2016, pp. 2741–2749.
[41] D. Bahdanau, K. Cho, Y. Bengio, Neural machine translation by jointly learning to align and translate, in: ICLR, 2015.
[42] T. Tieleman, G. Hinton, Lecture 6.5—RmsProp: Divide the gradient by a running average of its recent magnitude (2012).
[43] D. P. Kingma, J. Ba, Adam: A method for stochastic optimization, CoRR.
[44] P. Bojanowski, E. Grave, A. Joulin, T. Mikolov, Enriching word vectors with sub-word information, Transactions of the Association for Computational Linguistics 5 (2017) 135–146.
[45] E. Grave, P. Bojanowski, P. Gupta, A. Joulin, T. Mikolov, Learning word vectors for 157 languages, in: Proceedings of the International Conference on Language Resources and Evaluation, 2018.
[46] S. R. Kumar, G. Klaus, S. Jrgen, Highway networks, CoRR abs/1505.00387.