# DEEP ERROR-CORRECTING OUTPUT CODES

**Guoqiang Zhong**
Department of Computer Science and Technology
Ocean University of China
gqzhong@ouc.edu.cn

**Yuchen Zheng**
Department of Computer Science and Technology
Ocean University of China
ouczyc@outlook.com

**Peng Zhang**
Department of Computer Science and Technology
Ocean University of China
sdrzbruce@163.com

**Mengqi Li**
Department of International Trade and Economy
Ocean University of China
enri9615@outlook.com

**Junyu Dong**
Department of Computer Science and Technology
Ocean University of China
dongjunyu@ouc.edu.cn

## ABSTRACT

Existing deep networks are generally initialized with *unsupervised* methods, such as random assignments and greedy layerwise pre-training. This may result in the whole training process (initialization/pre-training + fine-tuning) to be very time-consuming. In this paper, we combine the ideas of ensemble learning and deep learning, and present a novel deep learning framework called deep error-correcting output codes (DeepECOC). DeepECOC are composed of multiple layers of the ECOC module, which combines multiple binary classifiers for feature learning. Here, the weights learned for the binary classifiers can be considered as weights between two successive layers, while the outputs of the combined binary classifiers as the outputs of a hidden layer. On the one hand, the ECOC modules can be learned using given *supervisory* information, and on the other hand, based on the ternary coding design, the weights can be learned only using part of the training data. Hence, the supervised pre-training of DeepECOC is in general very effective and efficient. We have conducted extensive experiments to compare DeepECOC with traditional ECOC, feature learning and deep learning algorithms on several benchmark data sets. The results demonstrate that DeepECOC perform not only better than traditional ECOC and feature learning algorithms, but also state-of-the-art deep learning models in most cases.

## 1 INTRODUCTION

Error correcting output codes (ECOC) are an ensemble learning framework to address multi-class classification problems (Dietterich & Bakiri, 1995). The work by (Zhong & Liu, 2013) shows that the ECOC methods can also be used for feature learning, in either a linear or a nonlinear manner. However, although sophisticated coding and decoding strategies are applied (Escalera et al., 2010; Zhong et al., 2012; Zhong & Cheriet, 2013), the learnability of ECOC is limited by its single-layer structure. Therefore, to exploit the advantages of the ECOC framework, such as supervised ensemble learning and effective coding design, it's necessary to combine its ideas with that of deep learning.

In recent years, many deep learning models have been proposed to handle various challenging problems. Meantime, desirable performances in many domains have been achieved, such as image classification and detection, document analysis and recognition, natural language processing, and video analysis (Hinton & Salakhutdinov, 2006; Krizhevsky et al., 2012; Szegedy et al., 2014; Simonyan & Zisserman, 2014; Zhang et al., 2015; Wang & Ji, 2015; Hong et al., 2015). Among others, (Hinton &

Salakhutdinov, 2006) presents the ground-breaking deep autoencoder that learns the weight matrices by pre-training the stacked restricted Boltzmann machines (RBMs) and fine-tuning the weights using gradient descent. It delivers much better representations of data than shallow feature learning algorithms, such as principal components analysis (PCA) (Jolliffe, 1986) and latent semantic analysis (LSA) (Deerwester et al., 1990). In order to boost the traditional autoencoder and prevent the "overfitting" problem, (Vincent et al., 2008) introduces the denosing autoencoder that corrupted the data with a random noise. Recently, most of the research focuses on deep convolutional neural networks (CNNs) and recurrent neural networks (RNNs), which greatly improves the state-of-the-art in the areas of object recognition, unsegmented handwriting recognition and speech recognition (Krizhevsky et al., 2012; Graves et al., 2009; Sak et al., 2014). However, existing deep networks are generally initialized with unsupervised methods, such as random assignments and greedy layerwise pre-training. In the case of random initialization, to obtain good results, many training data and a long training time are generally used; while in the case of greedy layerwise pre-training, as the whole training data set needs to be used, the pre-training process is very time-consuming and difficult to find a stable solution.

To overcome the limitations of both traditional ECOC methods and deep learning models, and meanwhile, take advantages of both of them, in this paper, we propose a novel deep learning model called deep error-correcting output codes (DeepECOC). DeepECOC are composed of multiple stacked ECOC modules, each of which combines multiple binary classifiers for feature learning. Here, the weights learned for the binary classifiers can be considered as weights between two successive layers, while the probabilistic outputs of the combined binary classifiers as the outputs of a hidden layer or new representations of data. On the one hand, the ECOC modules can be learned layer by layer using the given supervisory information, and on the other hand, based on the ternary coding design, some classes of data are automatically neglected when training the binary classifiers, such that the weights are learned only using part of the training data. Hence, the supervised pre-training of DeepECOC is in general very effective and efficient. We have compared DeepECOC with traditional ECOC, feature learning and deep learning algorithms to demonstrate the effectiveness and superiority of DeepECOC. The results are reported in Section 4.

The rest of this paper is organized as follows: In Section 2, we give a brief overview to related work. In Section 3, we present the proposed model, DeepECOC, in detail. The experimental results are reported in Section 4, while Section 5 concludes this paper with remarks and future work.

## 2 Related Work

Traditional ECOC framework has two steps: coding and decoding. In the coding step, an ECOC matrix is defined or learned from data, and the binary classifiers are trained based on the ECOC coding; in the decoding step, the class label is given to a test sample based on a similarity measure between codewords and outputs of the binary classifiers. The widely used coding strategies include one-versus-all (OneVsAll) (Nilsson, 1965), one-versus-one (OneVsOne) (Hastie et al., 1998), discriminant ECOC (DECOC) (Pujol et al., 2006), ECOC optimizing node embedding (ECOCONE) (Escalera et al., 2006), dense and sparse coding (Escalera et al., 2009; Allwein et al., 2001), and so on. Among them, the OneVsAll, OneVsOne, dense and sparse coding strategies are problem-independent, whilst the DECOC and ECOCONE are problem-dependent. Generally, the length of the codeword by the OneVsAll, OneVsOne, DECOC and ECOCONE coding designs is related to the number of classes, but that by the dense and sparse coding design is relatively flexible. In this work, we design the structure of DeepECOC based on the properties of each coding strategy. The commonly used binary ECOC decoding strategies are the Hamming decoding (Nilsson, 1965) and Euclidean decoding (Hastie et al., 1998). For ternary ECOC decoding strategies, the attenuated Euclidean decoding (Pujol et al., 2008), loss-based decoding (Allwein et al., 2001), and probabilistic-based decoding (Passerini et al., 2004) are widely used. Currently, the state-of-the-art ternary ECOC decoding strategies are the discrete pessimistic beta density distribution decoding and loss-weighted decoding (Escalera et al., 2010). In this work, for the simplicity of back propagation, we directly add a Softmax layer at the top of DeepECOC for the decoding. Note that, although many sophisticated coding and decoding strategies have been proposed in recent years (Escalera et al., 2010; Zhong et al., 2012; Zhong & Cheriet, 2013), the learnability of ECOC is limited by its single-layer structure. To further exploit the advantages of ECOC, such as supervised ensemble learning and effective coding design, it's necessary to combine its ideas with that of deep learning.


(a) one-versus-all (b) one-versus-one

Figure 1: Two coding matrices encoded with the one-versus-all (binary case) and one-versus-one (ternary case) coding strategies.

In the literature of deep learning, there is some work that attempts to construct a deep architecture with multiple feature learning methods (Hinton & Salakhutdinov, 2006; Trigeorgis et al., 2014; Yuan et al., 2015; Zheng et al., 2015; 2014). For instance, deep autoencoder is built up by RBMs (Hinton & Salakhutdinov, 2006), and deep semi-NMF combines multiple steps of matrix factorization (Trigeorgis et al., 2014). Similarly, deep CNNs and RNNs can also be considered as deep models that learn the new representations of data layer by layer (Krizhevsky et al., 2012; Graves et al., 2009; Sak et al., 2014). The success of these existing models demonstrate that deep networks are beneficial to the representation learning tasks, especially for the large scale applications. However, as discussed in the previous section, existing deep learning models are generally initialized with unsupervised methods, such as random assignments and greedy layerwise pre-training, which result in a long training time of the deep models. In this work, we propose the DeepECOC model, which is based on the stacked ECOC modules. When pre-training DeepECOC, the ECOC modules can be learned with the available supervisory information. Intuitively, as this manner of supervised pre-training has deterministic objective, the learned value of the parameters will be very close to the best local minimum on the solution manifold. Experimental results shown in Section 4 also demonstrate this fact.

## 3 DEEP ERROR-CORRECTING OUTPUT CODES (DEEPECOC)

In this section, we first introduce the traditional ECOC framework, which is the important building block of DeepECOC. Then we present the learning procedures of DeepECOC in detail.

### 3.1 THE ECOC FRAMEWORK

Error correcting output codes (ECOC), which combine multiple binary classifiers to solve multi-class classification problems, are an ensemble learning framework. The ECOC methods in general consist of two steps: coding and decoding. In the coding step, the ECOC coding matrix $\mathbf{M} \in \{-1, 1\}^{C \times L}$ (binary case) or $\mathbf{M} \in \{-1, 0, 1\}^{C \times L}$ (ternary case) is first defined or learned from the training data, where each row of $\mathbf{M}$ is the codeword of a class, each column corresponds to a dichotomizer (binary classifier), $L$ is the length of the codewords (the number of binary classifiers), $C$ is the number of classes, symbol '1' indicates positive class, '-1' indicates negative class, and '0' indicates that a particular class is not considered by a given classifier. Then, the binary classifiers (dichotomizers) are trained according to the partition of the classes in the columns of $\mathbf{M}$. Fig. 1 shows two coding matrices encoded with the one-versus-all (binary case) and one-versus-one (ternary case) coding strategies. The matrix is coded using several dichotomizers for a 4-class problem with respective codewords $\{y_1, \ldots, y_4\}$. The white girds are coded by 1 (considered as positive class by the respective dichotomizer $h_j$), the dark girds by -1 (considered as the negative class), and the gray girds by 0 (classes that are not considered by the respective dichotomizer $h_j$). In the decoding step, the test data are predicted based on an adopted decoding strategy and the outputs of the binary classifiers.

In order to take the probabilistic outputs of the base classifiers as new representations of data, we adopt linear support vector machines (linear SVMs) as the binary classifiers (dichotomizers), which solve a quadratic programming problem

$$\begin{aligned} \min_{\mathbf{w}, b, \xi_i} \; & J(\mathbf{w}) = \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{N}\xi_i \\ s.t. \quad & y_i f(\mathbf{x}_i) \geq 1 - \xi_i, \xi_i \geq 0, i = 1, \ldots, N \end{aligned} \tag{1}$$

where $\mathbf{w}$ and $b$ are the coefficients and bias of the binary classifier, $y_i \in \{+1, -1\}$, $\xi_i$'s are the slack variables, and $N$ is the number of the training data. The discriminant function can be expressed as

$$f(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + b. \tag{2}$$

This problem can be solved as,

$$\mathbf{w} = \sum_{i=1}^{N} \alpha_i y_i x_i, \tag{3}$$

$$b = \frac{1}{N_{\mathcal{SV}}} \sum_{\mathbf{x}_i \in \mathcal{SV}, i=1}^{N_{\mathcal{SV}}} (y_i - \mathbf{w}^T\mathbf{x}_i), \tag{4}$$

where $\alpha_i$'s are the non-negative Lagrange multipliers, $N_{sv}$ is the number of support vectors and $\mathcal{SV}$ is the set of support vectors. The dual form of Problem (1) can be written as

$$\begin{aligned} \max_{\alpha} \quad & \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{N} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j \\ = \quad & \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{N} \alpha_i \alpha_j y_i y_j k(\mathbf{x}_i, \mathbf{x}_j) \\ s.t. \quad & 0 \leq \alpha_i \leq \lambda, i = 1, \ldots, N, \\ & \sum_{i=1}^{N} \alpha_i y_i = 0, \end{aligned} \tag{5}$$

where $k(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i^T \mathbf{x}_j$ is the linear kernel function, $\lambda$ is a constant number, and $\alpha = \{\alpha_1, \ldots, \alpha_N\}$ is the vector of Lagrange multipliers. Replacing the linear kernel function with a nonlinear kernel, such as the Gaussian kernel

$$k(\mathbf{x}_i, \mathbf{x}_j) = \exp(-\sigma^{-1} \|\mathbf{x}_i - \mathbf{x}_j\|^2), \tag{6}$$

we can learn a nonlinear SVM, where $\sigma$ is the parameter for the Gaussian kernel function. The discriminant function of SVMs with a nonlinear kernel can be written as

$$f(\mathbf{x}) = \sum_{i=1}^{N} \alpha_i y_i k(\mathbf{x}_i, \mathbf{x}) + b. \tag{7}$$

Applying a decoding strategy on the outputs of the binary classifiers, the ECOC framework can be used for multi-class learning, while applying the sigmoid function on the values of the discriminant function, ECOC can be used for feature learning (Zhong & Liu, 2013). This is also the foundation of the DeepECOC model.

### 3.2 DEEPECOC

To combine the advantages of ECOC and deep learning algorithms, we build the DeepECOC architecture as follows

$$\mathbf{x} \xrightarrow{q_D} \tilde{\mathbf{x}} \xrightarrow[\mathbf{b}_1]{\mathbf{W}_1} \mathbf{h}_1 \xrightarrow[\mathbf{b}_2]{\mathbf{W}_2} \cdots \xrightarrow[\mathbf{b}_{n-1}]{\mathbf{W}_{n-1}} \mathbf{h}_{n-1} \xrightarrow{softmax} \mathbf{y}, \tag{8}$$

where the first step makes the clean input $\mathbf{x} \in [0, 1]^d$ partially destroyed by means of a stochastic mapping $\tilde{\mathbf{x}} \sim q_D(\tilde{\mathbf{x}} \mid \mathbf{x})$. In the corrupting process, we set a parameter called denoising rate $\nu$. For each input $\mathbf{x}$, a fixed number $\nu d$ of components are chosen at random, and their value is forced to 0, while the others are left untouched. This operation makes the model more robust and prevent the overfitting problem in most cases (Vincent et al., 2008). Subsequently, the "corrupted" data are taken as inputs for the DeepECOC model. $\mathbf{W}_1$ and $\mathbf{b}_1$ are the weight matrix and bias vector learned from the first ECOC module. The output of the first hidden layer is denoted as

$$\mathbf{h}_1 = s(\mathbf{W}_1^T \mathbf{x} + \mathbf{b}_1), \tag{9}$$

where $s(\cdot)$ is the sigmoid activation function $s(x) = \frac{1}{1+e^{-x}}$. From the second layer to the $(n-1)$-th layer, we use the stacked ECOC modules to learn the weight matrices and biases, which can be considered as weights between two successive layers of a deep network. Similarly, we use the output of the $(k-1)$-th layer as the input of the $k$-th layer,

$$\mathbf{h}_k = s(\mathbf{W}_k^T \mathbf{h}_{k-1} + \mathbf{b}_k). \tag{10}$$

Here, $\mathbf{h}_k$ can be viewed as an activation output and a new representation of the input datum $\mathbf{x}$.

For example, if we adopt the OneVsAll coding strategy for one layer of the ECOC module, we first define the coding matrix $\mathbf{M}^{C\times C}$, where $C$ is the number of classes. Then, we can train $C$ SVM classifiers to obtain the weight matrix $\mathbf{W} = \{\mathbf{w}_1, \ldots, \mathbf{w}_i, \ldots, \mathbf{w}_C\}$ and the bias $\mathbf{b} = \{b_1, \ldots, b_i, \ldots, b_C\}$. Next, we calculate the output of the first layer by using Eq. (9). Subsequently, we repeat this process layer by layer to build the DeepECOC model. It's obvious that, if we adopt different coding strategies, we can get different kinds of DeepECOC architectures.

For the last layer of DeepECOC, we employ the softmax regression for the multi-class learning. Its cost function is defined as

$$J(\mathbf{w}) = -\frac{1}{N}\left(\sum_{i=1}^{N}\sum_{j=1}^{K} \mathbf{I}(y_i = j) \log \frac{\exp(\mathbf{w}_j^T \mathbf{h}_i^{n-1})}{\sum_{l=1}^{K} \exp(\mathbf{w}_l^T \mathbf{h}_i^{n-1})}\right), \tag{11}$$

where $\mathbf{I}(x)$ is the indicator function, $\mathbf{I}(x) = 1$ if $x$ is true, else $\mathbf{I}(x) = 0$. $y_i$ is the label corresponding to $\mathbf{x}_i$. It's easy to compute the probability that $\mathbf{x}_i$ is classified to class $j$,

$$p(y_i = j|\mathbf{x}_i, \mathbf{w}) = \frac{\exp(\mathbf{w}_j^T \mathbf{h}_i^{n-1})}{\sum_{l=1}^{K} \exp(\mathbf{w}_l^T \mathbf{h}_i^{n-1})}. \tag{12}$$

Taking derivatives, one can show that the gradient of $J(\mathbf{w})$ with respect to $\mathbf{w}$ is,

$$\nabla J(\mathbf{w}) = -\frac{1}{N}\sum_{i=1}^{N}[\mathbf{x}_i(\mathbf{I}(y_i = j) - p(y_i = j|\mathbf{x}_i, \mathbf{w}))]. \tag{13}$$

After the pre-training step, we use back propagation (Rumelhart et al., 1988) to fine tune the whole architecture. Moreover, we also employ a technique called "dropout" for regularization (Hinton et al., 2012). When a large feedforward neural network is trained on a small training set, dropout generally performs well on the test set. The basic idea of dropout is that each hidden node is randomly omitted from the network with a probability of $\beta$. In another view, dropout is a very efficient way to perform model averaging with neural networks. Through these processes, we finally obtain the DeepECOC model, which is robust and easy to be applied to multi-class classification tasks.

Note that, compared to existing deep learning algorithms, DeepECOC have some important advantages. Firstly, unlike previous deep learning algorithms, DeepECOC are built with the ECOC modules and pre-trained in a supervised learning fashion. Secondly, if we adopt ternary coding strategies, due to the natural merit of ECOC, the weights can be learned using only part of the training data. Thirdly, in contrast to the learning of the weight matrices in previous deep learning models, the binary classifiers in each ECOC module can be learned in parallel, which may greatly speed up the learning of DeepECOC.

## 4 EXPERIMENTS

To evaluate the effectiveness of the proposed method, DeepECOC, we conducted 4 parts of experiments. In the first part, we compared DeepECOC with some deep learning models and single-layer ECOC approaches on 16 data sets from the UCI machine learning repository [1]. In the second part, we compared DeepECOC with traditional feature learning models, some deep learning models and single-layer ECOC approaches on the USPS handwritten digits [2], and tested DeepECOC with different number of hidden layers. In the third part, we used the MNIST handwritten digits [3] to further

[1] http://archive.ics.uci.edu/ml/

[2] http://www-i6.informatik.rwth-aachen.de/~keysers/usps.html

[3] http://yann.lecun.com/exdb/mnist/

Table 1: Details of the UCI data sets (T: training samples; A: attributes; C: classes).

| Problem | ♯ of T | ♯ of A | ♯ of C | Problem | ♯ of T | ♯ of A | ♯ of C |
|---|---|---|---|---|---|---|---|
| Dermatology | 366 | 34 | 6 | Yeast | 1484 | 8 | 10 |
| Iris | 150 | 4 | 3 | Satimage | 6435 | 36 | 7 |
| Ecoli | 336 | 8 | 8 | Letter | 20000 | 16 | 26 |
| Wine | 178 | 13 | 3 | Pendigits | 10992 | 16 | 10 |
| Glass | 214 | 9 | 7 | Segmentation | 2310 | 19 | 7 |
| Thyroid | 215 | 5 | 3 | Optdigits | 5620 | 64 | 10 |
| Vowel | 990 | 10 | 11 | Shuttle | 14500 | 9 | 7 |
| Balance | 625 | 4 | 3 | Vehicle | 846 | 18 | 4 |

Table 2: Classification accuracy and standard deviation obtained by DeepECOC and the compared approaches on 16 UCI data sets. Here, DeepECOC(1)∼ DeepECOC(3) are 3 variant of DeepECOC with the ECOCONE coding design initialized by one-versus-one, one-versus-all and DECOC respectively. The best results are highlighted in boldface.

| **Problem** | Single | AE | DAE | DeepECOC (1) | DeepECOC (2) | DeepECOC (3) |
|---|---|---|---|---|---|---|
| Dermatology | 0.9513 | 0.9429 ±0.0671 | 0.9674 ±0.0312 | 0.9702 ±0.0354 | **0.9779 ±0.0208** | 0.9747 ±0.0318 |
| Iris | **0.9600** | **0.9600 ±0.0562** | 0.9333 ±0.0889 | **0.9600 ±0.0535** | 0.9267 ±0.1109 | 0.9533 ±0.0383 |
| Ecoli | 0.8147 | 0.7725 ±0.0608 | 0.8000 ±0.0362 | 0.8529 ±0.0403 | 0.8824 ±0.0626 | **0.9118 ±0.0636** |
| Wine | 0.9605 | 0.9765 ±0.0264 | 0.9563 ±0.0422 | **0.9875 ±0.0264** | 0.9813 ±0.0302 | 0.9688 ±0.0329 |
| Glass | 0.6762 | 0.6669 ±0.1032 | 0.6669 ±0.0715 | **0.7895 ±0.0788** | 0.7368 ±0.1140 | 0.7562 ±0.0879 |
| Thyroid | 0.9210 | 0.9513 ±0.0614 | 0.9599 ±0.0567 | 0.9656 ±0.0513 | **0.9703 ±0.0540** | 0.9608 ±0.0518 |
| Vowel | 0.7177 | 0.6985 ±0.0745 | 0.7101 ±0.0756 | **0.7475 ±0.0901** | 0.6010 ±0.0627 | 0.6863 ±0.0788 |
| Balance | 0.8222 | 0.8036 ±0.0320 | 0.8268 ±0.0548 | 0.9137 ±0.0412 | 0.8333 ±0.0318 | **0.9167 ±0.0312** |
| Yeast | 0.5217 | 0.5641 ±0.0346 | 0.5891 ±0.0272 | **0.5959 ±0.0599** | 0.5494 ±0.0434 | 0.5697 ±0.0462 |
| Satimage | 0.8537 | 0.8675 ±0.0528 | 0.8897 ±0.0304 | 0.8961 ±0.0480 | 0.8360 ±0.0390 | **0.9077 ±0.0555** |
| Letter | 0.9192 | 0.9234 ±0.0547 | 0.9381 ±0.0641 | **0.9532 ±0.0341** | 0.9247 ±0.0352 | 0.9501 ±0.0563 |
| Pendigits | 0.9801 | 0.9831 ±0.0123 | 0.9886 ±0.0034 | **0.9908 ±0.0031** | 0.9866 ±0.0107 | 0.9899 ±0.0075 |
| Segmentation | 0.9701 | 0.9584 ±0.0317 | 0.9596 ±0.0211 | **0.9711 ±0.0286** | 0.9584 ±0.0163 | **0.9711 ±0.0233** |
| Optdigits | **0.9982** | 0.9785 ±0.0101 | 0.9856 ±0.0088 | 0.9867 ±0.0096 | 0.9848 ±0.0123 | 0.9911 ±0.0091 |
| Shuttle | 0.9988 | 0.9953 ±0.0012 | 0.9976 ±0.0014 | 0.9988 ±0.0021 | 0.9983 ±0.0018 | **0.9993 ±0.0010** |
| Vehicle | 0.7315 | 0.6987 ±0.0521 | 0.7348 ±0.0454 | **0.7561 ±0.0480** | 0.6908 ±0.04321 | 0.7195 ±0.0148 |
| Mean rank | 4.0938 | 4.8750 | 3.9375 | **1.7500** | 3.9375 | 2.4063 |

demonstrate the effectiveness of DeepECOC for handwritten digits recognition. Finally, the CIFAR-10 data set [4] was used to demonstrate the effectiveness of DeepECOC on image classification tasks. For all the data sets, the features were normalized within $[0, 1]$. In the following, we report the experimental results in detail.

### 4.1 Classification on 16 UCI Machine Learning Repository Data Sets

The detail of the UCI data sets are shown in Table 1. In these experiments, we compared DeepECOC with autoencoder (AE) (Hinton & Salakhutdinov, 2006), denoising autoencoder (DAE) (Vincent et al., 2008) and single-layer ECOC approaches (Single) (Escalera et al., 2010). We built DeepECOC with the ECOC optimizing node embedding (ECOCONE) coding method (Escalera et al., 2006). Here, since we initialized ECOCONE with 3 different coding methods, i.e. one-versus-one, one-versus-all and DECOC, DeepECOC had 3 variants. In addition, the state-of-the-art linear loss-

[4] http://www.cs.toronto.edu/∼kriz/cifar.html

weighted (LLW) decoding strategy was used for ECOCONE. Finally, a structure with 3 hidden layers was adopted for DeepECOC, which had 0.1 denoising rate and 0.1 dropout rate:

$$\mathbf{x} \xrightarrow{q_D} \tilde{\mathbf{x}} \xrightarrow[\mathbf{b}_1]{\mathbf{W}_1} \mathbf{h}_1 \xrightarrow[\mathbf{b}_2]{\mathbf{W}_2} \mathbf{h}_2 \xrightarrow[\mathbf{b}_3]{\mathbf{W}_3} \mathbf{h}_3 \xrightarrow{softmax} \mathbf{y}. \tag{14}$$

For the fine-tuning process, we used the stochastic gradient descent algorithm. The learning rate and epoches from different data sets are described in Table 3. The autoencoder and denoising autoencoder's architectures are as same as DeepECOC with ECOCONE initialized by one-versus-one. For single-layer ECOC approaches, we chose the best results shown in (Escalera et al., 2010) as our compared results. For all DeepECOC models, we used support vector machines (SVMs) with RBF kernel function as base classifiers. The parameters of SVMs were set to default (Chang & Lin, 2011).

Table 2 shows the average classification accuracy and standard deviation on 16 UCI data sets. Except on the OptDigits data set, DeepECOC achieved the best results compared with autoencoder, denoising autoencoder and single-layer ECOC approaches. In fact, on the OptDigits data set, DeepECOC achieved comparative result with single-layer ECOC approaches. Among others, DeepECOC with ECOCONE (initialized by one-versus-one) coding strategy obtained the best results on 9 data sets, while DeepECOC with ECOCONE (initialized by DECOC) coding strategy obtained the best results on 5 data sets. From the mean rank values, we can see that DeepECOC with ECOCONE (initialized by one-versus-one and DECOC) strategy far surpass other compared methods.

Table 3: Details of the learning rate $\eta$ and epoch on the UCI data sets.

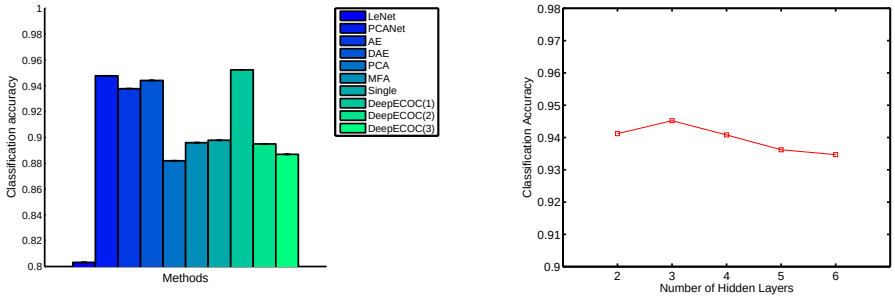
| Problem | $\eta$ | Epoch | Problem | $\eta$ | Epoch |
|---|---|---|---|---|---|
| Dermatology | 0.1 | 2000 | Yeast | 0.01 | 4000 |
| Iris | 0.1 | 400 | Satimage | 0.01 | 4000 |
| Ecoli | 0.1 | 2000 | Letter | 0.01 | 8000 |
| Wine | 0.1 | 2000 | Pendigits | 0.01 | 2000 |
| Glass | 0.01 | 4000 | Segmentation | 0.01 | 8000 |
| Thyroid | 0.1 | 800 | Optdigits | 0.01 | 2000 |
| Vowel | 0.1 | 4000 | Shuttle | 0.1 | 2000 |
| Balance | 0.1 | 4000 | Vehicle | 0.1 | 4000 |

### 4.2 Classification on the USPS data set

The USPS handwritten digits data set includes 7291 training samples and 2007 test samples from 10 classes. The size of the images is $16 \times 16 = 256$. Our experiments on this data set were divided into 2 parts. Firstly, we compared DeepECOC with two traditional feature learning models (principal components analysis (PCA) (Jolliffe, 2002) and marginal Fisher analysis (MFA) (Yan et al., 2007)), autoencoder (AE), denoising autoencoder (DAE), LeNet (LeCun et al., 1998), PCANet (Chan et al., 2015) and single-layer ECOC approaches. Here, PCA is an unsupervised method, MFA is a supervised method. For MFA, the number of nearest neighbors for constructing the intrinsic graph was set to 5, while that for constructing the penalty graph was set to 15. For DeepECOC, we also used 3 coding design methods in this experiment. We used batch gradient descent for the fine-tuning process, the batch size was set to 100, the learning rate was set to 1, the number of epoch was set to 40000, the denoising rate, and dropout rate were set to 0.1. We also used SVMs with RBF kernel and default parameters as base classifiers. For single-layer ECOC approaches, we adopted ECOCONE (initialized by one-versus-one) as coding design method and linear loss-weighted (LLW) decoding strategy. For the LeNet model, we used 2 convolutional layers, two pooling layers and two fully connected layers. The kernel size of the convolutional layers and pooling layers was set to $2 \times 2$, the stride was set to 1, the number of nodes of the first layer was set to 200, the epoch was set to 8000, the initial learning rate was set to 0.001, learning rate policy was set to "inv", and the momentum was set to 0.9. For the PCANet model, we used two PCA-filter stages, one binary hashing stage and one blockwise histograms. The filter size, the number of filters, and the block size were set to $k_1 = k_2 = 3$, $L_1 = L_2 = 4$, and $7 \times 7$, respectively. The experimental results are shown in Fig. 2(a).

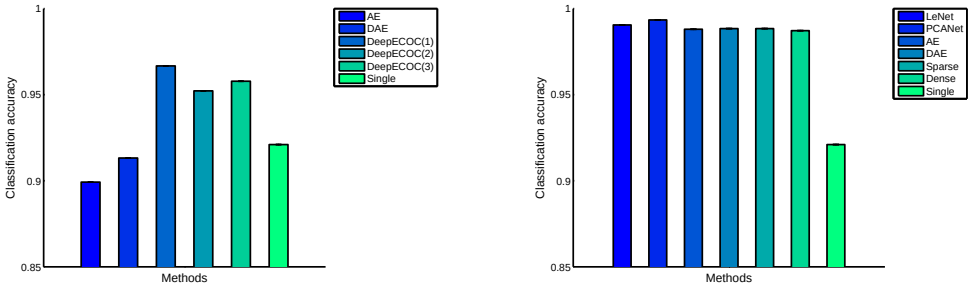
From Fig. 2(a), we can see that DeepECOC with ECOCONE (initialized by one-versus-one) coding strategy achieved the best result than other methods include traditional feature learning models, existing deep learning methods and single-layer ECOC approaches.

In the second part, we evaluated DeepECOC with different number of hidden layers. We used 2 to 6 hidden layers in our experiments. The parameter settings were as same as the first part. Fig. 2(b)




(a) (b)

Figure 2: (a) Classification accuracy obtained on the USPS data set. Here, DeepECOC(1)$\sim$ DeepECOC(3) are 3 variant of DeepECOC with the ECOCONE coding design initialized by one-versus-one, one-versus-all and DECOC respectively. (b) Classification accuracy with different numbers of hidden layers on the USPS data set.




(a) $784 - Z_1 - Z_2 - Z_3 - 10$ (b) $784 - 500 - 500 - 2000 - 10$

Figure 3: Classification accuracy obtained on the MNIST data set for two architectures.

shows the experimental results. We can see that DeepECOC obtained the best result when using 3 hidden layers. When the number of hidden layers is less than 3, the effectiveness of DeepECOC increases with the increasing of the number of hidden layers. Along with the number of hidden layers continues to grow, the effectiveness of DeepECOC decreases.

### 4.3 Classification on the MNIST data set

MNIST handwritten digits data set has a training set of 60,000 examples, and a test set of 10,000 examples with 784 dimensional features. We designed 2 architectures for autoencoder, denoising autoencoder and DeepECOC. The first architecture was $784 - Z_1 - Z_2 - Z_3 - 10$, where $Z_i$ was the number of hidden neurons designed based on some ECOC coding strategies. We designed this architecture because we wanted to make autoencoder and denoising autoencoder had the same structure with DeepECOC. The second architecture is $784-500-500-2000-10$. This architecture was used in (Hinton & Salakhutdinov, 2006). In order to make DeepECOC adapt to this structure, we used the dense and sparse coding design methods that can control the codeword length. Note that, the dense and sparse coding design methods are totally random and data-independent. The denoising rate and dropout rate were set to 0.1, the batch size was set to 100, the learning rate was set to 0.01, and the number of epoch was set to 80000. For LeNet model, we adopted the parameters as same as (LeCun et al., 1998). For PCANet model, we used two PCA-filter stages, one binary hashing stage and one blockwise histograms. In the PCANet, the filter size, the number of filters, and the block size were set to $k_1 = k_2 = 8$, $L_1 = L_2 = 7$, and $7 \times 7$, respectively.

Fig. 3(a) and Fig. 3(b) show the experimental results on 2 architectures. We can see that DeepECOC are comparative with existing deep learning methods on the second architecture and outperform

Table 4: Classification accuracy obtained on the LBP-CIFAR10 data set. The best result for each scenario is highlighted in bold face.

| Problem | AE | DAE | LeNet | PCANet | DeepECOC(1) | DeepECOC(2) | DeepECOC(3) |
|---|---|---|---|---|---|---|---|
| LBP-CIFAR10 (36) | 0.3501 | 0.3678 | 0.3256 | 0.2569 | **0.5089** | 0.4517 | 0.4752 |
| LBP-CIFAR10 (256) | 0.4352 | 0.4587 | 0.3221 | 0.2569 | **0.5588** | 0.4589 | 0.5224 |

them on the first architecture. In addition, DeepECOC with both two architectures outperform the single-layer ECOC approaches.

### 4.4 Classification on the LBP-CIFAR10 Data Set

The CIFAR-10 dataset is a relative large scale data set which consists of 60000 $32 \times 32$ colour images in 10 classes, with 6000 images per class. There are 50000 training images and 10000 test images. For the purpose of reducing computational cost, we attempted to extract features of the data using an efficient local binary patterns algorithm. As a result, the representations with dimensionality 36 and 256 were adopted and the data were normalized to [0, 1] as well, called LBP-CIFAR10 (36) and LBP-CIFAR10 (256). We also used 3 hidden layers for all deep learning methods. The learning rate was set to 0.1, and the epoch was set to 4000. For the LeNet model, we used 2 convolutional layers and two fully connected layers without pooling layers. The kernel size was set to $2 \times 2$, the stride was set to 1, the number of node of the first fully connected layer was set to 64, the epoch was set to 4000, the initial learning rate was set to 0.01, learning rate policy was set to "inv", and the momentum was set to 0.9. For the PCANet model, we used two PCA-filter stages, one binary hashing stage and one blockwise histograms. In the PCANet, the filter size, the number of filters, and the block size were set to $k_1 = k_2 = 3$, $L_1 = L_2 = 4$, and $7 \times 7$, respectively. The classification accuracy are reported in Table 4.

From Table 4, we can easy to see that DeepECOC achieved the best results. Moreover, DeepECOC with ECOCONE (initialized by one-versus-one) coding strategy achieved the better results than autoencoder and denoising autoencoder, LeNet and PCANet. Hence, we can conclude that, Deep-ECOC are a general model to handle different real world applications and achieves desirable results in most cases.

## 5 Conclusion

In this paper, we propose a novel deep learning model, called deep error correcting output codes (DeepECOC). DeepECOC extend traditional ECOC algorithms to a deep architecture fashion, and meanwhile, brings new elements to the deep learning area, such as supervised initialization, and automatic neglecting of part of the data during network training. Extensive experiments on 16 data sets from the UCI machine learning repository, the USPS and MNIST handwritten digits and the CIFAR-10 data set demonstrate the superiority of DeepECOC over traditional ECOC, feature learning and deep learning methods. In future work, we will further exploit the learnability of DeepECOC on large scale applications.

## References

E. L. Allwein, R. E. Schapire, and Y. Singer. Reducing multiclass to binary: A unifying approach for margin classifiers. *The Journal of Machine Learning Research*, 1:113–141, 2001.

T.-H. Chan, K. Jia, S. Gao, J. Lu, Z. Zeng, and Y. Ma. PCANet: A Simple Deep Learning Baseline for Image Classification? *Image Processing, IEEE Transactions on*, 24(12):5017–5032, 2015.

C.-C. Chang and C.-J. Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2(3):27, 2011.

S. Deerwester, S. Dumais, T. Landauer, G. Furnas, and R. Harshman. Indexing by Latent Semantic Analysis. *JASIS*, 41(6):391–407, 1990.

T. G. Dietterich and G. Bakiri. Solving multiclass learning problems via error-correcting output codes. *Journal of artificial intelligence research*, pp. 263–286, 1995.

S. Escalera, O. Pujol, and P. Radeva. Ecoc-one: A novel coding and decoding strategy. In *ICPR*, volume 3, pp. 578–581, 2006.

S. Escalera, O. Pujol, and P. Radeva. Separability of ternary codes for sparse designs of error-correcting output codes. *Pattern Recognition Letters*, 30(3):285–297, 2009.

S. Escalera, O. Pujol, and P. Radeva. On the decoding process in ternary error-correcting output codes. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 32(1):120–134, 2010.

A. Graves, M. Liwicki, S. Fernandez, R. Bertolami, H. Bunke, and J. Schmidhuber. A Novel Connectionist System for Unconstrained Handwriting Recognition. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 31(5):855–868, 2009.

T. Hastie, R. Tibshirani, et al. Classification by pairwise coupling. *The annals of statistics*, 26(2): 451–471, 1998.

G. Hinton and R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, 2006.

G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*, 2012.

S. Hong, T. You, S. Kwak, and B. Han. Online tracking by learning discriminative saliency map with convolutional neural network. In *ICML*, 2015.

I. Jolliffe. *Principal Component Analysis*. New York: Springer-Verlag, 1986.

I. Jolliffe. *Principal component analysis*. Wiley Online Library, 2002.

A. Krizhevsky, I. Sutskever, and G. Hinton. ImageNet classification with deep convolutional neural networks. In *NIPS*, pp. 1106–1114, 2012.

Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

N. J. Nilsson. *Learning Machines*. McGraw-Hill, 1965.

A. Passerini, M. Pontil, and P. Frasconi. New results on error correcting output codes of kernel machines. *Neural Networks, IEEE Transactions on*, 15(1):45–54, 2004.

O. Pujol, P. Radeva, and J. Vitria. Discriminant ECOC: a heuristic method for application dependent design of error correcting output codes. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 28(6):1007–1012, 2006.

O. Pujol, S. Escalera, and P. Radeva. An incremental node embedding technique for error correcting output codes. *Pattern Recognition*, 41(2):713–725, 2008.

D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning representations by back-propagating errors. *Cognitive modeling*, 5:3, 1988.

H. Sak, A. Senior, and F. Beaufays. Long short-term memory recurrent neural network architectures for large scale acoustic modeling. In *INTERSPEECH*, pp. 338–342, 2014.

K. Simonyan and A. Zisserman. Very Deep Convolutional Networks for Large-Scale Image Recognition. *CoRR*, abs/1409.1556, 2014.

C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going Deeper with Convolutions. *CoRR*, abs/1409.4842, 2014.

G. Trigeorgis, K. Bousmalis, S. Zafeiriou, and B. Schuller. A deep semi-nmf model for learning hidden representations. In *ICML*, pp. 1692–1700, 2014.

P. Vincent, H. Larochelle, Y. Bengio, and P.-A.Manzagol. Extracting and composing robust features with denoising autoencoders. In *ICML*, pp. 1096–1103, 2008.

X. Wang and Q. Ji. Video Event Recognition with Deep Hierarchical Context Model. In *CVPR*, pp. 4418–4427, 2015.

S. Yan, D. Xu, B. Zhang, H.-J. Zhang, Q. Yang, and S. Lin. Graph embedding and extensions: a general framework for dimensionality reduction. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 29(1):40–51, 2007.

Y. Yuan, L. Mou, and X. Lu. Scene Recognition by Manifold Regularized Deep Learning Architecture. *Neural Networks and Learning Systems, IEEE Transactions on*, 26(10):2222–2233, 2015.

X. Zhang, J. Zhao, and Y. LeCun. Character-level convolutional networks for text classification. In *NIPS*, pp. 649–657, 2015.

Y. Zheng, G. Zhong, J. Liu, X. Cai, and J. Dong. Visual texture perception with feature learning models and deep architectures. In *Pattern Recognition*, pp. 401–410. Springer, 2014.

Y. Zheng, Y. Cai, G. Zhong, Y. Chherawala, Y. Shi, and J. Dong. Stretching deep architectures for text recognition. In *ICDAR*, pp. 236–240, 2015.

G. Zhong and M. Cheriet. Adaptive error-correcting output codes. In *IJCAI*, 2013.

G. Zhong and C.-L. Liu. Error-correcting output codes based ensemble feature extraction. *Pattern Recognition*, 46(4):1091–1100, 2013.

G. Zhong, K. Huang, and C.-L. Liu. Joint learning of error-correcting output codes and dichotomizers from data. *Neural Computing and Applications*, 21(4):715–724, 2012.