IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received May 5, 2021, accepted May 31, 2021, date of publication June 11, 2021, date of current version June 22, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3088717

Deep Convolutional Neural Network Ensembles Using ECOC

SARA ATITO ALI AHMED[1,2], (Member, IEEE), CEMRE ZOR[3], MUHAMMAD AWAIS[2], BERRIN YANIKOGLU[1,4], (Senior Member, IEEE), AND JOSEF KITTLER[2], (Life Member, IEEE)

[1]Faculty of Engineering and Natural Sciences, Sabanci University, 34956 Istanbul, Turkey
[2]Centre for Vision, Speech and Signal Processing (CVSSP), University of Surrey, Guildford 7XH, U.K.
[3]Centre for Medical Image Computing (CMIC), University College London, London 6BT, U.K.
[4]Center of Excellence in Data Analytics, Sabanci University, 34956 Istanbul, Turkey

Corresponding author: Sara Atito Ali Ahmed (saraatito@sabanciuniv.edu)

This work was supported in part by TÜBİTAK, The Scientific and Technological Research Council of Turkey, under Project 119E429.

ABSTRACT Deep neural networks have enhanced the performance of decision making systems in many applications, including image understanding, and further gains can be achieved by constructing ensembles. However, designing an ensemble of deep networks is often not very beneficial since the time needed to train the networks is generally very high or the performance gain obtained is not very significant. In this paper, we analyse an error correcting output coding (ECOC) framework for constructing ensembles of deep networks and propose different design strategies to address the accuracy-complexity trade-off. We carry out an extensive comparative study between the introduced ECOC designs and the state-of-the-art ensemble techniques such as ensemble averaging and gradient boosting decision trees. Furthermore, we propose a fusion technique, that is shown to achieve the highest classification performance.

INDEX TERMS Deep learning, ensemble learning, error correcting output coding, gradient boosting decision trees, multi-task classification.

I. INTRODUCTION

Classifier ensembles are a popular method to boost the performance of a classification system. The combination rules employed for fusing ensembles of base classifiers can be as simple as taking a vote, or more complex, involving learning to compensate for the respective weaknesses of the individual base classifiers.

Several fusion techniques such as averaging, majority voting [1], bagging [2], stacking [3], random forests [4], error correcting output coding [5], [6] and their variants have been widely used in traditional machine learning. However, extensions of some of these approaches to deep learning (DL) systems have been deemed inefficient and challenging, due to the computational complexity associated with the training of deep networks, as well as the difficulty in ensuring diversity among the base classifiers. Therefore, most of the state-of-art DL ensembles are either formed of simple averaging (or voting) frameworks, comprising only a small number of base classifiers [7]–[11], or weak decision tree ensembles based on boosting deep features that have been already extracted [12]–[15].

Averaging ensembles are composed of base classifiers that are mainly obtained by modifying the various DL elements such as the network architectures, and their parameters, data augmentation techniques, and the meta parameters of the learning process. An example is the DeepFace [7], where Taigman *et al.* construct a face verification system of 7 deep networks and achieve 97.35% accuracy, compared to 97.0% obtained using a single face verification network. In another work, Szegedy *et al.* [8] increase the accuracy from 40% (single network) to 43.9% by averaging 6 GoogLeNet networks in the ILSVRC 2015 detection challenge. Yet another example is the winner of the PlantCLEF2017 competition [16], which is formed of 12 networks that are trained with an emphasis on complementarity and achieved a top-1 accuracy of 88.5% in classifying 10,000 different plant species. Similarly, Gessert *et al.* [11] employ multi-resolution EfficientNets [17] for skin lesion classification based on an ensemble of 15 deep networks, where the area under the curve (AUC) is increased from an average of 94 per classifier to 95.4.

Despite the performance gain achieved by the deep averaging ensembles, the increased time complexity, which scales

The associate editor coordinating the review of this manuscript and approving it for publication was Liviu-Adrian Cotfas.

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/

linearly with the addition of each base classifier network, comes out as the main drawback. In the literature, gradient boosting decision tree (GBDT) methods are proposed to address this shortcoming, by operating on the deep features obtained from one base network (contrary to generating many deep networks as base classifiers) and constructing a sequential ensemble of trees which are trained to correct each other's errors, using these features.

There are three commonly used GBDT variations in the literature: extreme gradient boosting (XGBoost) [18], light gradient boosting machine (LightGBM) [19], and categorical boosting (CatBoost) [20]. As an example of XGBoost, Pang *et al.* [14] propose a subcellular localisation method by integrating the Convolutional Neural Network (CNN) and XGBoost, where CNN acts as a feature extractor and XGBoost acts as a classifier to identify the protein subcellular localisation. In another literature review, Torres-Barrán *et al.* [21] study the application of XGBoost to global and local wind energy prediction and solar radiation problem, exploiting gradient boosting regression methods. As for LightGBM, Ju *et al.* [15] overcome the limitation of the single-convolution model in predicting the wind power by integrating the LightGBM algorithm to improve the robustness and accuracy of the forecasting.

Although GBDT is a powerful ensemble technique, the major disadvantages are its inability to deal with a high number of classes and the high number of hyper-parameters that need to be tuned to obtain the desired performance. It is important to note that the improved time complexity obtained with respect to the averaging ensembles is at the expense of a reduced ensemble performance. In this article, we address the drawbacks of deep averaging ensembles (time complexity) and GBDT (accuracy), and propose an efficient DL framework based on error correcting output coding (ECOC).

ECOC, borrowed originally from the communication theory [22], [23], is a multi-class classification ensemble, in which a given multi-class problem is decomposed into several two-class problems, whose simpler decision boundaries are then combined to give the final, more complex decision boundary. The errors of the base classifiers that implement the two-class decision boundaries are corrected to a certain degree [24]. Several data-dependent and independent approaches can be used for guiding the decomposition process [25], [26]. In [27], it has been theoretically and experimentally demonstrated that ECOC frameworks formed using random class splits obtain close to Bayes performance, if there are infinitely many such splits, and the associated base classifiers achieve good accuracy. In practice, the performance reaches the optimum very rapidly [28] as the number of classifiers increases. The superiority of this method, which we refer to as *randECOC*, over the rest of the data independent and dependent ECOC approaches are demonstrated in [24], [27], and [29].

Although ECOC has been commonly employed in traditional machine learning applications [30]–[33], to date, to the best of our knowledge, its potential as a method of constructing deep convolutional neural network ensembles has neither been exploited nor analysed. The only work addressing ECOC in DL research is [34], where it is utilised for the adversarial robustness of the networks.

In this work, by operating on the base network features, we propose and analyse efficient implementation strategies for randECOCs. We investigate three different design procedures: i) the straightforward approach of training the base classifiers independently, ii) multi-task learning (MTL) for faster training, and iii) MTL with embedded error correction. It is expected that the selection of the most appropriate design procedure will be carried out by the user, depending on the specific requirements of an application, and the time complexity versus accuracy trade-off.

The systems proposed are evaluated on four public datasets: CIFAR-10 and CIFAR-100 (10 and 100 classes, respectively) for object recognition [35], SVHN of Google street view images of house numbers (10 classes) [36], and PlantVillage dataset [37] consisting of 38 plant leaf disease types.

We show that for all the proposed design techniques, randECOC almost always surpasses the GBDT performance at comparable time complexity, when MTL based implementation strategies are considered. When compared with averaging ensembles, a degradation in performance has been noted, due to the end-to-end training nature of averaging ensembles as opposed to the feature-based training of randECOC. However, for the users who have enough resources to accommodate averaging ensembles, we propose combining randECOC with averaging, and show that this setup guarantees the best performance with the highest accuracy in all scenarios.

The main contributions of this study can be summarised as follows:

- We propose three different designs for randECOC ensembles to be used with convolutional deep neural networks, and analyze these approaches in terms of accuracy and time complexity, using several different deep network architectures and 4 different datasets.
- We perform an empirical comparison of the randECOC ensembles and state-of-the-art ensemble methods for deep learning, i.e. ensemble averaging and GBDTs, and show that the proposed MTL strategies provide the best time complexity versus accuracy trade-off.
- We propose a hybrid approach, combining randECOC strategies and ensemble averaging, to achieve state-of-the art classification performance for all network and dataset combinations.

The article is structured as follows. Section II provides a background information on the state-of-the-art deep ensemble classification techniques as well as the ECOC framework. In Section III, different ECOC training strategies using features extracted by deep convolutional neural networks are presented. This is followed by their experimental analysis in Section IV and a discussion of the results obtained. Finally, conclusions of this study are presented in Section V.

II. BACKGROUND

In the literature, averaging and majority voting are the most commonly used classifier combination approaches, where the ensemble output is calculated based on the (weighted) average of the base classifier outputs or their mostly voted prediction. Bagging [2] is a special case of majority voting for which the base classifiers are trained on different versions [38] of the same data obtained by resampling, to ensure complementarity among the base classifiers. More complex combination rules include methods such as boosting [39], where the classifiers are trained sequentially to compensate for the weaknesses of those already selected and stacking [40], where the outputs of all classifiers are fed into a new model to generate the final prediction. Another commonly used ensemble technique is random forests [4], which are composed of multiple decision trees trained on bootstrapped training data with an additional step of feature-bootstrapping to allow for a random selection (with replacement) of features at each tree node. The final decision is based on the (weighted) average of the outputs or the majority vote of the individual tree decisions.

The fusion rules most commonly applied in the state-of-the-art deep learning ensembles are based on averaging or majority voting. These ensembles consist of a small number of deep neural network architectures as base classifiers, which differ from each other in terms of the data augmentation techniques used during training and / or network architectures and / or learning parameters (such as learning rates, training and validation set partitions, weights initialisation and data batches). Due to the costly training of these ensembles, they typically are composed of only a handful of base classifiers.

Overcoming the time complexity of the averaging / voting ensembles of deep neural networks, the second most common combination strategy, gradient boosting decision trees (GBDT), depends on extracting the bottleneck features of one *base network* and using them for training a sequence of decision trees. However, the gain in time complexity of this approach is compromised by reduced accuracy, especially for high number of classes. Moreover, the method requires a high number of hyper-parameters to be tuned to obtain the desired performance.

In this work, we confine the comparison of our results to that of simple averaging ensembles of deep neural networks and gradient boosting methods; we do not include bagging or stacking as they both involve training multiple deep networks, which takes a very long time. Furthermore, while bagging might bring additional benefits over simple averaging ensembles, especially for smaller data sets, this is beside the point, as our experimental validation of the proposed methods is based on the fact that their results *approach* that of simple averaging ensembles, while having much smaller time complexity.

In Section II-A, we analyse three state-of-the-art variants of the GBDT method found in the literature; namely, extreme gradient boosting (XGBoost) [18], light gradient boosting machine (LightGBM) [19], and categorical boosting (CatBoost) [20], in detail. In Section II-B, we provide the background for error correcting output coding (ECOC) ensembles, on which we build our novel design strategies for designed ensembles of deep learning networks, presented in Section III.

A. GRADIENT BOOSTING DECISION TREES (GBDT)

Gradient boosting is a machine learning technique for regression and classification problems that creates an ensemble of weak prediction models to achieve powerful prediction. When decision trees are used as the base classifiers, the method is referred to as gradient boosting decision trees (GBDT).

Unlike random forests, where the decision trees are constructed in parallel prior to combination, GBDT employs a boosting approach, in which each tree is sequentially trained with the aim of correcting the error produced by its predecessor. In particular, every tree is trained to learn the residual between the desired output and the output of the previous tree, using gradient descent.The most important parameter in GBDT is the number of base classifiers which controls the model complexity. The most recent and efficient GBDT methods developed are XGBoost [18], LightGBM [19], and CatBoost [20]. These algorithms differ from each other in terms of the mechanism used for splitting the tree nodes.

Extreme gradient boosting (XGBoost) [18], is a highly extensible tool mainly designed to overcome the overfitting limitations of the traditional gradient boosting methods. It uses pre-sorted and histogram-based algorithms for computing the best split, which continues until the maximum level, pre-defined by the ''max_depth'' hyper-parameter, is reached. Once at the maximum level, the splits are pruned backwards until there is no positive gain.

Light gradient boosting machine (LightGBM), proposed and developed by Microsoft [19], uses gradient-based one-side sampling (GOSS) to filter out data instances on the basis of their contribution to the gradient of the loss function. The best split is obtained by using all of the instances with large gradients and a random sample of instances with small gradients to maintain a balance between the training data reduction and accuracy. LightGBM uses a leaf-wise tree growth mechanism which allows the growth of an imbalanced tree.

Categorical boosting (CatBoost) [20] focuses on categorical features by using minimal variance sampling (MVS), which is a weighted sampling method at the tree-level. Unlike LightGBM, CatBoost grows balanced trees, which makes this method less prone to overfitting, and uses combinations of categorical features as additional categorical features to capture high-order dependencies. As it is infeasible to process all of the possible combinations, CatBoost solves the exponential growth of the feature combinations by constructing the candidates in a greedy way.

B. ERROR CORRECTING OUTPUT CODING (ECOC)

Error Correcting Output Coding (ECOC) is a generic ensemble classification framework designed for multi-class classification problems [24], where the aim is to decompose a given multi-class problem into several two-class problems. The final decision boundary is formed by combining the boundaries of the base classifiers trained on these simple decompositions, while providing a scope for error correction.

The way the decomposition is carried out in ECOC is defined by a *design code matrix*. Accordingly, a base classifier may be assigned the task of separating a particular class from all of the others, or learning a random dichotomy of the classes. The commonly used ensemble approaches such as one-vs-one or one-vs-all can therefore be considered as special types of ECOC systems.

Let us consider a problem with K classes $\{c_1, c_2, \ldots c_K\}$, L base classifiers $\{h_1, h_2, \ldots h_L\}$, and a pre-designed code matrix $\mathbf{M}$ of size $K \times L$ as illustrated in Table (1), for $K = 4$ and $L = 5$. A particular element $M_{ij} \in \{+1, -1\}$ indicates the desired label for class c_i to be used in training the base classifier, h_j. For instance in Table (1), the base classifier, h_1, is assigned the task of separating instances belonging to classes c_1 and c_2 from instances belonging to classes c_3 and c_4. The classes c_1 and c_2 are re-labelled with label $+1$, while c_3 and c_4 are re-labelled with label -1, to reflect this two-class problem.

TABLE 1. A sample ECOC matrix for a 4-class classification problem with 5 base classifiers.

	h_1	h_2	h_3	h_4	h_5
c_1	+1	+1	+1	-1	-1
c_2	+1	-1	-1	+1	-1
c_3	-1	+1	-1	+1	-1
c_4	-1	-1	-1	-1	+1

The design (encoding) of the code matrix can be carried out in several ways. These include problem-independent approaches such as one-vs-one or one-vs-all [24], or problem-dependent methodologies where the aim is to split the classes in the given data domain [29], [41] meaningfully.

In decision making (testing), firstly, a given test instance $\mathbf{x}$ is classified by each base classifier to obtain the output vector $\mathbf{Y} = [y_1, \ldots, y_L]$ where y_j is the hard or soft output of the classifier h_j for $\mathbf{x}$. Then, the distance between $\mathbf{Y}$ and the *codeword* $\mathbf{M_i}$ of class c_i, $\forall i$, is computed using a metric such as Hamming, Manhattan or Euclidean distance. The class $c*$ associated with the minimum distance is chosen as the predicted class, such that

$$c* = \arg \min_{i=1\ldots k} \text{dist}(\mathbf{Y}, \mathbf{M_i}) \tag{1}$$

While choosing the closest codeword during the target prediction, the system is able to correct some of the base classifiers mistakes. Specifically, up to $\lfloor (e-1)/2 \rfloor$ base classifier errors can be corrected if Hamming Distance (HD) is chosen as the distance metric, and the minimum HD between any pair of codewords is e.

Although the encoding and decoding of ECOC matrices are open research problems, it is important to note that randomly generated ECOC matrices (randECOC) have been shown to reach Bayes performance when used with a large enough number of base classifiers, each of which is exhibiting close to Bayes accuracy [27]. In practice, it has been experimentally demonstrated in [28] that for problems involving $\sim$10 classes, randECOCs of length 20-30 would be enough to converge to optimum performance, whereas this number would grow to 200-300, when the number of classes is $\sim$100.

III. DESIGN STRATEGIES FOR randECOC USING CNNs

Under the assumption of unconstrained computational resources, the optimal strategy to achieve the highest prediction performance using randECOC would be to train each base classifier independently. End-to-end training of these classifiers, each of which is initialised with random weights, would help increase the diversity between classifiers and enforce independence which is a key element in achieving close-to-Bayes performance [27], [28]. However, this procedure would suffer from similar time complexity drawbacks as in averaging ensembles and be impractical in real-life applications.

For this reason, in this section, we propose and analyse different design strategies for randECOC matrices, which address the shortcomings of time complexity associated with averaging ensembles, while still achieving better performance than their time efficient alternative, GBDT. In the design strategies presented in Section III-A through III-C, we propose to initially train a multi-class *base network* to obtain the bottleneck features (as opposed to end-to-end training), and build three implementation techniques with varying accuracy vs time complexity trade-off on these features.

Specifically, after presenting the straightforward approach to designing randECOC ensembles with base classifiers trained independently using bottleneck features in Section III-A, we propose a more time-efficient implementation strategy based on multi-task learning (MTL) in Section III-B. Then, in Section III-C, the MTL based strategy is further improved with the incorporation of an error-correcting mechanism as a separate layer of the network. This strategy aims to couple the base classifier training to the classification problem, as opposed to training the base classifiers only to be in agreement with the encoding matrix: A few research works exist to learn or modify the ECOC matrix after the training of the base classifiers, for their joint optimization [42]–[44].

In our study, due to resource constraints, we confine the choice of base networks to convolutional neural networks (CNNs). However, it cannot be overemphasised that the proposed ensemble design methodology is general and would be just as applicable to other deep neural network architectures.

A. INDEPENDENT LEARNING OF BASE CLASSIFIERS

In this approach, the base classifiers are trained one by one and independently according to a given randECOC matrix,

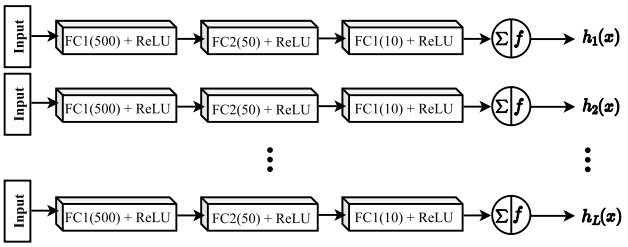

FIGURE 1. An independent base classifier architecture with a 3-hidden layer shallow network, consisting of fully connected layers followed by rectified linear units, one for each base classifier of the ECOC ensemble. The input comprises the features extracted by the bottleneck layer of a trained *base network*.

using the deep features extracted from the bottleneck layer of a base network. A schematic diagram illustrating an example of independently trained base classifier networks is given in Figure 1.

Here, we propose to design the base classifiers as shallow networks, whose outputs are then combined for an error-correcting randECOC decoding to give the final output. In other words, after extracting the output vector $\mathbf{Y}(\mathbf{x})$ for a given test sample $\mathbf{x}$ from all shallow networks, the prediction is carried out in a *separate* decoding step, where $\mathbf{x}$ is assigned the class with the closest codeword to $\mathbf{Y}(\mathbf{x})$ (see Equation 1).

B. MULTI-TASK LEARNING OF BASE CLASSIFIERS

In order to achieve close-to-Bayes accuracy, the number of base classifiers required for a randECOC ensemble should increase with the number of classes. Although all independent tasks can potentially be trained in parallel as proposed in Section III-A, this framework might be unattractive under the assumption of limited resources, despite the performance gain promised.

To address this, we consider the idea of simultaneous training of the base classifiers by employing an MTL based strategy, where the classifiers are trained to learn multiple labels, i.e. the desired base classifier outputs, at the same time. Although this method can only approximate the performance of the independently trained base classifiers, it is important from the point of view of accuracy versus time complexity trade-off.

In this approach, we have a single MTL network comprising several shared layers among all base classifiers, with L output nodes, as opposed to L independent networks. In other words, while training the independent classifiers sequentially would mean the repetition of the randECOC procedure L times, training all classifiers at the same time via MTL would imply carrying out this step only once. Hence, the time complexity of the MTL network is approximately L times better than the independent sequential training. An illustration of an example MTL network is presented in Figure 2.

The prediction is carried out in the same way as in Section III-A, where ECOC decoding is executed as the second step, following the extraction of classifier outputs in the first step. Note that we propose that this network should also include a small number of shallow, classifier specific layers to allow for diversity.

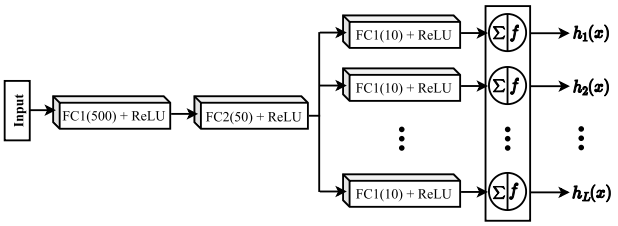

FIGURE 2. Multi-task learning architecture, with two shared modules and one classifier specific module. All layers are fully connected networks with rectified linear units.

As a further advantage of the MTL network, it should be noted that the sharing of the base network and the subsequent layers are expected to reduce overfitting, as observed in the literature [45], since the nodes in the shared layers are constrained to work for multiple classifiers.

C. MULTI-TASK LEARNING WITH EMBEDDING

Despite its advantages in terms of speed and reduced overfitting, the MTL network described in Section III-B is suboptimal in the sense that the second step of the prediction, namely ECOC decoding, is carried out separately from the network training. In other words, while the base classifiers are enforced to learn the dichotomies (two-class problems) indicated by the randECOC matrix, they are not enforced to reveal the desired *multi-class* label.

In order to address this issue, we propose to extend the MTL network with a K-node output layer, with weights set from the randECOC codewords and the output nodes representing the original classes. This layer not only enforces the final, multi-class decision on the outputs of the two-class base classifiers, but also inherently includes the ECOC decoding. The proposed framework is illustrated in Figure 3 with an example architecture. It is referred to as "MTL w/ embedding" in the remainder of this paper.

It is worth mentioning that the randECOC matrix is not learned here but is pre-set. In some earlier work, the matrix was modified during or after the training of the base classifiers, with the goal of reducing this decoupling between the encoding and base classifier training stages [42]–[44].

Let us assume that the nodes corresponding to the base classifiers $h_j, j = 1 \ldots L$ are connected to the output nodes $o_i, i = 1 \ldots K$ with the preset ECOC matrix weights $w_{ij} = M_{ij}$. For a given input $\mathbf{x}$, each output node o_i represents the score for class c_i, such that

$$o_i(\mathbf{x}) = \sum_{j=1}^{L} h_j(\mathbf{x}) \times w_{ij} = \mathbf{h}(\mathbf{x}) \cdot \mathbf{w_i}. \tag{2}$$

Note that the maximum value of $o_i(\mathbf{x})$ is L when all the base classifier outputs are in agreement with their associated bits of the codeword for that class (targets); while the minimum is $-L$ when all base classifier outputs are wrong.

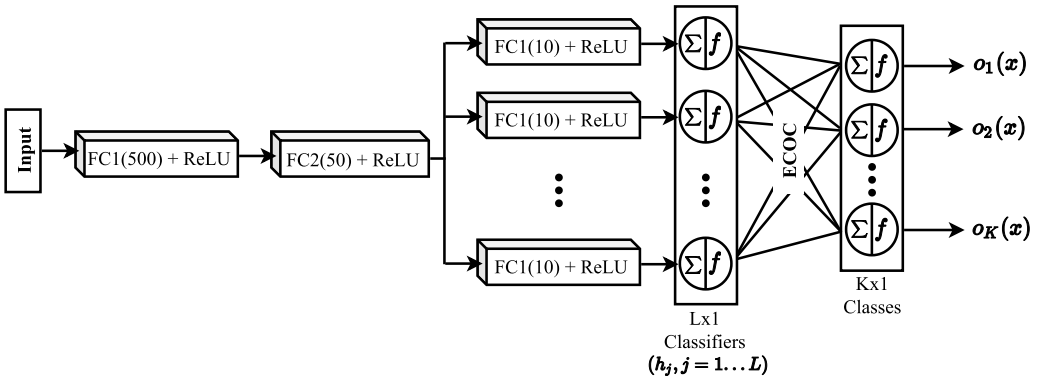

FIGURE 3. Multi-label architecture with embedded ECOC decoding, including two shared modules and one classifier specific module. The base classifier output layer is followed by the ECOC embedding layer with fixed weights. The output o_i corresponds to the score of class c_i.

In other words,

$$HD(\mathbf{w_c}, \mathbf{h}(\mathbf{x})) = \frac{L - o_c(\mathbf{x})}{2}. \tag{3}$$

The loss function used to train the network is designed with two goals: 1) To maximise the output of the correct class, o_c; 2) To match the output vector $\mathbf{h}(\mathbf{x})$ to the predetermined codeword $\mathbf{w_c}$, so as to benefit from the ECOC framework. Therefore, given a sample of class c and groundtruth $\mathbf{T} = [t_1 \ldots t_K]$ (one-hot encoded vector where $t_c = 1$ for only the correct class and zero elsewhere), we use the loss function given in Equation 4. We ignore $o_i, i \neq c$ because maximizing o_c is equivalent to minimizing other class outputs, thanks to the design of the ECOC matrix.

$$\mathcal{L} = (L - o_c(\mathbf{x}))^2 + \sum_{l=1}^{L} (h_l(\mathbf{x}) - M(c, l))^2 \tag{4}$$

With ternary ECOC where there are zeros in the code matrix, the maximum output value of L is not attainable for o_c, hence L should be replaced with the number of non-zeros in a codeword.

To train the network, we use stochastic backpropagation, starting with the weights of the base classifiers h_j, as the ECOC matrix weights are fixed. The partial derivative of our combined loss function with respect to $h_j(\mathbf{x})$ is computed as:

$$\begin{aligned}
&\partial \mathcal{L} / \partial h_j(x) \\
&= \frac{\partial (L - o_c(x))^2}{\partial o_c(x)} \frac{\partial o_c(x)}{\partial h_j(x)} + \sum_{l=1}^{L} \frac{\partial (h_l(x) - M(c, l))^2}{\partial h_j(x)} \\
&= 2(L - o_c(x))\, \omega_{cj} + 2\left(h_j(x) - M(c, j)\right)
\end{aligned}$$

For the final prediction, the class c_i that has the maximum $o_i(\mathbf{x})$ (equivalently, minimum distance to the base classifier outputs $\mathbf{h}(\mathbf{x})$) is chosen as the correct class.

IV. EXPERIMENTAL ANALYSIS AND RESULTS

To evaluate the effectiveness of the proposed randECOC techniques and compare their efficiency in terms of time complexity and accuracy with the state-of-the-art ensemble methods, we conduct various experiments using well-known deep architectures and multi-class datasets. Specifically, the comparative studies are performed on:

1) Simple averaging ensemble;
2) Gradient boosting decision trees (GBDTs): XGBoost, LightGBM, and CatBoost;
3) randECOC ensembles: Independent learning, MTL, and MTL with embedding.

After carrying out the comparisons, we combine randECOC and GBDT approaches with ensemble averaging, i.e. we generate *ensembles* of randECOC and GBDT ensembles and analyse their performance. The purpose of this experiment is to measure the highest possible prediction accuracy, for scenarios where the available resources (computational resources including processing power, time and storage) are not a limiting factor for the user.

In Section IV-A, the details of the datasets used in the experiments are presented and in Section IV-B, various base network architectures utilised in this study are described. This is followed by providing the details of the experimental setup in Section IV-C, and the thorough discussion of the results in Section IV-D.

A. DATASETS

We carry out the experimental analyses on four state-of-the-art multi-class classification problems based on digit classification and object recognition using images. In all tasks, each image contains a single object on an unconstrained background.

- **CIFAR-10 [35]:** This dataset consists of 60, 000 (32×32) images belonging to 10 classes (airplane, automobile, bird, cat, deer, dog, frog, horse, ship, and truck), and is divided into 50, 000 images to be used for training and 10, 000 for testing.
- **CIFAR-100 [35]:** Similar to CIFAR-10 dataset, CIFAR-100 consists of 50, 000 training images and 10, 000 testing images. There are 100 classes in this dataset, grouped into 20 super-classes. Each image comes with a

"fine label" which is the class label and a "coarse label" which is the super-class to which it belongs. In our study, we make use of the fine labels.

- **SVHN [36]:** This real-world dataset comprises house numbers obtained from Google Street View images and consists of 73, 257 samples for training, 26, 032 images for testing and 531, 131 additional, less difficult samples which can be used as extra data for training. In our study, the training portion of the dataset corresponding to isolated digits (10 classes) is used.
- **PlantVillage [37]:** This crowd-sourced dataset consists of 54, 309 images with 39 diseases of different crop plants. Three different versions of this dataset are provided: original RGB images with varied sizes, gray-scaled version of the raw images, and RGB images with just the leaf segmented and color corrected. In this work, we used the original RGB images.

B. BASE NETWORK

To construct the base network, we employ four commonly used, state-of-the-art convolutional neural network architectures; namely MobileNetV2 [46], Inception-V3 [47], Xception [48], and Squeeze-and-Excitation Networks (SENet) [49].

MobileNetV2 proposed by Google in [46], is a lightweight deep neural networks where depthwise separable convolution is used to reduce the model size and complexity. The network is 53 layers deep with a total of 3.54 million parameters.

Inception-V3, proposed by Google in [47], is a widely-used image recognition model. It consists of symmetric and asymmetric building blocks, including convolutions, average pooling, max pooling, dropouts, and fully connected layers. Batch normalisation is used extensively throughout the model and applied to activation inputs. The network is 48 layers deep with a total of 23.8 million parameters.

Xception [48] is an extreme version and an extension of the Inception [8] architecture, which replaces the standard inception modules with depth-wise separable convolutions. The network is 71 layers deep with a total of 22.9 million parameters.

The final architecture, SENet [49], introduces the squeeze-and-excitation block that adaptively re-calibrates the channel-wise feature responses by modelling the interdependencies between channels to automatically acquire the importance of each feature channel. SENet is the winner of ILSVRC 2017 classification challenge.

C. EXPERIMENTAL SETTING

All the base networks employed in this study are pre-trained on the ImageNet dataset [50]. The inputs to the ensemble systems are obtained by forward passing the datasets through the fine-tuned networks and extracting the features of the last pooling layer. Note that for a fair comparison, no extra randomness such as data augmentation, has been applied during training or feature extraction.

The averaging ensembles are obtained by training 5 base networks with different random weight initialisations, as this is a typical number employed in these ensembles to respect time and computing power constraints. For randECOC using independently trained base classifiers, we train L classifiers with a simple multi-layer perceptron architecture, where L is set to 30, 300, 30, and 100 for CIFAR-10, CIFAR-100, SVHN, and PlantVillage datasets, respectively. The architecture, which is depicted in Figure 1, consists of three fully connected layers with (500, 50, 10) units, each followed by rectified linear unit (*ReLU*) activation function and a dropout layer. Finally, each output layer has one neuron that is associated with a tangent hyperbolic activation function (*tanh*) and a mean square error (*MSE*) loss function.

The randECOC framework using MTL is composed of two shared fully connected layers with (500, 50) units, each of which is followed by a *ReLU* activation function and a dropout layer. For each classifier, there are some specific layers: a dropout layer, a fully connected layer with 10 units, a *ReLU* activation function, a fully connected layer with one unit, and a *tanh* function. The output units are concatenated to form one layer with L units defining the output layer. Similar to randECOC with independently trained base classifiers, *MSE* loss function is used here. The network structure of this framework is as given in Figure 2.

The randECOC using MTL w/ embedding, as given in Figure 3, mimics the setup of the randECOC framework using MTL, with an additional layer to include ECOC codewords as weights, for which the learning rate is set to zero. Note that as the random weight initialisations impacts on training all randECOC frameworks, we report the mean and the standard deviation of the testing accuracy from 5 independent runs.

All the networks including the base networks are optimised using RMSPROP optimiser with 3×10^{-4} learning rate, 0.99 squared gradient decay factor, and a batch size of 64 images per training iteration for the base networks and 512 images for the randECOC experiments. The implementation is performed using the Deep Learning Toolbox and MatConvNet [51] within MATLAB, with a single NVIDIA GeForce GTX 1080 Ti 11GB graphics processing unit (GPU).

The GBDT frameworks are implemented using the official XGBoost, LightGBM and CatBoost Python packages on Google Colaboratory with the provided free Tesla K80 11GB GPU. In our experiments, we fine-tune the most vital hyper-parameter for these frameworks, which is the number of iterations that is relative to the number of created trees. The highest validation accuracy has been obtained using 60 iterations for CIFAR10 and SVHN datasets and 300 iterations for CIFAR100 in all of the employed gradient boosting methods. Rest of the hyper-parameters are set to the default values suggested by the corresponding authors.

For the set of experiments where the randECOC and GBDT frameworks are combined with ensemble averaging, we train 5 base networks, each of which is initialised by random weights, separately for all network architectures and dataset combinations. For each network architecture, we first

TABLE 2. Comparison of the results obtained on the CIFAR-10 dataset using MobileNetV2, Inception-V3, Xception, and SENet architectures as base networks. The best results obtained in each group are shown in bold and the performance decreases compared to the base networks are shown underlined. The numbers in parentheses show the performance change compared to the base network.

			Gradient Boosting Ensemble			randECOC Ensemble		
	Base Network	Ensemble of 5 base networks	XGBoost	LightGBM	CatBoost	Independent Classifiers	MTL	MTL w/ embedding
MobileNetV2								
Testing Accuracy	93.10%	96.01% (+2.91)	**93.43%** (+0.33)	93.01% (−0.09)	93.28% (+0.18)	**94.05%** ±0.022 (+0.95)	93.94% ±0.024 (+0.84)	93.98% ±0.025 (+0.88)
Training Time (in minutes)	273	1365	0.299	2.51	0.240	96.1	3.98	4.01
Testing Time (in minutes)	0.82	4.1	0.002	0.004	0.001	0.102	0.019	0.019
Inception-V3								
Testing Accuracy	93.56%	96.14% (+2.58)	**94.39%** (+0.83)	94.03% (+0.47)	93.37% (−0.19)	**94.61%** ±0.027 (+1.05)	94.45% ±0.033 (+0.89)	94.49% ±0.087 (+0.93)
Training Time (in minutes)	580	2,900	0.78	4.25	0.36	103	4.27	4.63
Testing Time (in minutes)	1.70	8.50	0.002	0.006	0.021	0.130	0.022	0.022
Xception								
Testing Accuracy	94.88%	97.02% (+2.14)	**95.18%** (+0.30)	95.13% (+0.25)	94.98% (+0.10)	**95.52%**±0.044 (+0.64)	95.40%±0.062 (+0.52)	95.42%±0.048 (+0.54)
Training Time (in minutes)	722	3,611	0.76	4.31	0.36	103	4.27	4.63
Testing Time (in minutes)	2.79	13.9	0.002	0.007	0.027	0.124	0.022	0.022
SENet								
Test Accuracy	95.93%	97.69 % (+1.76)	**96.33%** (+0.40)	96.12% (+0.19)	95.07% (−0.86)	**96.82%** ±0.024 (+0.89)	96.81%±0.067 (+0.88)	96.81%±0.074 (+0.88)
Training Time (in minutes)	1430	7,154	0.780	4.58	0.36	103	4.27	4.62
Testing Time (in minutes)	6.98	34.92	0.002	0.007	0.026	0.129	0.022	0.022

evaluate the ensemble averaging performance with the 5 networks. Then, GBDT and randECOC approaches are applied to the features extracted from each base network, resulting in 5 ensembles in each case. The ECOC matrices used in all 5 networks are kept the same. Finally, ensemble averaging is applied to the 5 GDBT and 5 randECOC ensembles to obtain the final prediction.

To further validate our results, we carry out a final set of experiments with the PlantVillage dataset which is a large, crowd-sourced dataset of real-life diseased plant images.

D. RESULTS

We report the result of a comparison of the evaluated frameworks in Section IV-D1 and the performance analysis of combinatory approaches in Section IV-D2.

1) COMPARISON OF THE ENSEMBLE FRAMEWORKS

The performance of the 5 ensemble frameworks together with their corresponding time complexity, while using four base networks, is shown in Table 2, 3, and 4 for CIFAR-10, CIFAR-100 and SVHN datasets, respectively. The performance is gauged in terms of classification accuracy, and the time complexity is measured as the training and test time spent, *over and above* the time required by the base network. Note that while the hardware is slightly different for GBDT and ECOC frameworks, their time complexities are both accepted as small and no strict comparison is made between the two in terms of time.

a: ENSEMBLE AVERAGING

As expected, the averaging ensemble achieves the highest accuracy for all datasets and base networks, with highest accuracies of 97.69%, 89.91% and 97.75% for the CIFAR-10, CIFAR-100 and SVHN datasets, respectively. Despite surpassing the base network by a relatively high margin, this ensemble comes out as very costly in terms of time and the resources required. Specifically, the training times are of the order of thousands of minutes, or about several days, which is often not available to researchers, providing the motivation for this work.

b: GRADIENT BOOSTING DECISION TREES

Among the GBDT frameworks, none outperforms the others on all datasets, and more importantly, it can be observed that all three variations cause degradation over the base network performance at least for network architecture and dataset. This is a very important finding, proving a clear evidence in support of the perceived instability and inconsistency of this technique, especially when dealing with a high number of classes. Specifically, for CIFAR-10 and SVHN datasets, XGBoost appears as the best performing algorithm as shown in Table 2 and 4, respectively. It improves the testing accuracy

TABLE 3. Comparisons on the CIFAR-100 dataset using the MobileNetV2, Inception-V3, Xception, and SENet architectures as base networks. The best results obtained in each group are shown in bold.

			Gradient Boosting Ensemble			randECOC Ensemble		
	Base Network	Averaging of 5 base networks	XGBoost	LightGBM	CatBoost	Independent Classifiers	MTL	MTL w/ embedding
MobileNetV2								
Testing Accuracy	74.61%	81.95% (+7.34)	**72.52%** (−2.09)	75.83% (+1.22)	75.04% (+0.43)	**77.28%** ±0.027 (+2.67)	76.65% ±0.021 (+2.04)	76.74% ±0.036 (+2.13)
Training Time (in minutes)	592	2960	0.310	2.48	0.310	98.5	4.10	4.23
Testing Time (in minutes)	0.85	4.6	0.002	0.006	0.008	0.106	0.018	0.019
Inception-V3								
Test Accuracy	76.77%	81.34% (+4.57)	73.67% (−3.10)	**76.47%** (−0.30)	76.12% (−0.65)	**78.91%** ±0.025 (+2.14)	78.34% ±0.091 (+1.57)	78.45% ±0.096 (+1.68)
Training Time (in minutes)	1160	5800	47.3	175	5.27	981	34.8	35.7
Testing Time (in minutes)	1.66	8.28	0.012	0.344	0.024	1.25	0.187	0.191
Xception								
Test Accuracy	80.67%	85.50% +4.83)	79.30% (−1.37)	**81.70%** (+1.03)	81.58% (+0.91)	**83.24%** ±0.035 (+2.57)	82.97% ±0.077 (+2.30)	83.16% ±0.081 (+2.49)
Training Time (in minutes)	1342	6709	47.7	178	5.35	1025	38.2	38.4
Testing Time (in minutes)	3.03	15.2	0.012	0.327	0.025	1.35	0.197	0.205
SENet								
Test Accuracy	87.35%	89.91% (+2.56)	84.17% (−3.18)	86.39% (−0.96)	**86.51%** (−0.84)	**87.90%** ±0.040 (+0.55)	87.60% ±0.037 (+0.25)	87.74% ±0.130 (+0.39)
Training Time (in minutes)	1463	7317	48.4	179	5.57	1005	35.8	36.4
Testing Time (in minutes)	7.67	38	0.007	0.391	0.045	1.28	0.195	0.213

TABLE 4. Comparisons on the SVHN dataset using the MobileNetV2, Inception-V3, Xception, and SENet architectures as base networks. The best results obtained in each group are shown in bold.

			Gradient Boosting Ensemble			randECOC Ensemble		
	Base Network	Averaging of 5 base networks	XGBoost	LightGBM	CatBoost	Independent Classifiers	MTL	MTL w/ embedding
MobileNetV2								
Testing Accuracy	95.20%	97.02% (+1.82)	**95.66%** (+0.46)	95.53% (+0.33)	95.07% (−0.13)	**95.92%** ±0.083 (+0.72)	95.88% ±0.082 (+0.68)	95.89% ±0.094 (+0.69)
Training Time (in minutes)	381	1905	0.309	3.82	0.512	103	6.04	5.98
Testing Time (in minutes)	0.95	6.51	0.005	0.012	0.008	0.204	0.032	0.038
Inception-V3								
Test Accuracy	95.63%	97.51% (+1.88)	**96.52%** (+0.89)	96.42% (+0.79)	96.20% (+0.57)	**96.76%** ±0.018 (+1.13)	96.67% ±0.017 (+1.04)	96.67% ±0.045 (+1.04)
Training Time (in minutes)	690	3,450	0.820	1.57	0.570	151	6.49	6.81
Testing Time (in minutes)	4.42	22.1	0.004	0.008	0.035	0.335	0.064	0.065
Xception								
Test Accuracy	96.83%	97.75% (+0.92)	**97.03%** (+0.20)	97.02% (+0.19)	96.98% (+0.15)	**97.15%** ±0.011 (+0.32)	97.10% ±0.010 (+0.27)	97.12% ±0.010 (+0.29)
Training Time (in minutes)	830	4,150	1.14	2.28	0.34	153	6.53	6.85
Testing Time (in minutes)	7.59	37.9	0.004	0.007	0.037	0.342	0.063	0.064
SENet								
Test Accuracy	94.70%	95.81% (+1.11)	**94.89%** (+0.19)	94.36% (+0.34)	92.85% (−1.85)	**95.81%** ±0.099 (+1.11)	95.72% ±0.015 (+1.02)	95.79% ±0.038 (+1.09)
Training Time (in minutes)	1,877	9,594	1.14	2.34	0.49	151	6.58	6.79
Testing Time (in minutes)	18.2	90.5	0.005	0.007	0.043	0.402	0.061	0.065

of the base networks at the expense of minimal additional training time, with improvements of 0.33%, 0.83%, 0.30%, and 0.40% on CIFAR-10 and 0.46%, 0.89%, 0.20%, and 0.19% on the SVHN dataset, compared to the base network.

TABLE 5. Test accuracies for the combinatory methods. The best result corresponding to each dataset and base network, is shown in bold.

			Gradient Boosting Ensemble			randECOC Ensemble		
	Base Network	Averaging of 5 base networks	XGBoost	LightGBM	CatBoost	Independent Classifiers	MTL	MTL w/ embedding
CIFAR-10								
MobileNetV2	93.10%	96.01%	95.86%	96.03%	95.90%	**96.17%**	96.09%	96.10%
Inception-V3	93.56%	96.14%	96.22%	96.25%	96.03%	**96.42%**	96.35%	96.35%
Xception	94.88%	97.02%	97.09%	97.06%	97.11%	**97.25%**	97.18%	97.20%
SENet	95.93%	97.69%	97.70%	97.77%	97.16%	**97.85%**	97.79%	97.84%
CIFAR-100								
MobileNetV2	74.61%	81.95%	80.52%	81.98%	81.67%	**82.83%**	82.47%	82.62%
Inception-V3	76.77%	81.34%	81.50%	82.50%	81.68%	**83.34%**	83.12%	83.26%
Xception	80.67%	85.50%	85.59%	85.88%	85.95%	**86.65%**	86.51%	86.55%
SENet	87.35%	89.91%	88.87%	89.30%	88.44%	**90.11%**	90.00%	90.07%
SVHN								
MobileNetV2	95.20%	97.02%	97.16%	97.18%	97.07%	**97.26%**	97.24%	97.25%
Inception-V3	95.63%	97.51%	97.57%	97.41%	97.56%	**97.59%**	97.55%	97.56%
Xception	96.83%	97.75%	97.74%	97.71%	97.72%	**97.80%**	97.75%	97.78%
SENet	94.70%	95.81%	96.14%	96.02%	95.67%	**96.27%**	96.16%	96.22%

However, this method deteriorates the base network accuracy for all network types on CIFAR-100. For this dataset, the only GBDT improvement over the base network performance is achieved when using Xception as the architecture and employing LightGBM or CatBoost. This is in line with the theoretical underpinning of the inability of these methods to cope with a high number of classes [52].

c: randECOC ENSEMBLES

We see that all the variants of the randECOC framework improve the testing accuracy over the base networks and the best GBDT approach,[1] in almost all of our experiments. Despite some drop in the performance in comparison to the averaging ensembles, a much faster training time is observed. For instance, in the case of CIFAR-100, the averaging ensemble requires around 2, 4, 4.6, and 5 days for training and reveals 81.95%, 81.34%, 85.50%, and 89.91% test accuracy, with different base network architectures. On the other hand, randECOC using MTL w/ embedding requires only about 30 minutes for the training of all the architectures, with the output test accuracy of 76.74%, 78.45%, 83.16%, and 87.74%. While MTL w/ embedding brings roughly half the performance improvement obtained by the averaging ensemble over the base network, it does so consistently and requiring negligible additional time, which is important for scenarios where training several deep networks is not viable.

Among the MTL based randECOC ensembles, MTL w/ embedding performs always better than or equal to MTL, while revealing similar time complexity. The independent learning approach obtains the highest accuracy; however only with a slight margin over MTL w/ embedding and a lot more additional training time (more than 20 times in all scenarios).

The strength of the MTL based randECOC approaches over GBDTs is emphasised especially when dealing with high number of classes. As shown in Table 3 for the CIFAR-100, MTL w/ embedding improves the accuracy by 2.13%, 1.68%, 2.49%, and 0.39% over the base networks, and outperforms the best GBDT approach (LightGBM in this case) by 0.91%, 1.98%, 1.46%, and 1.35%, for the four network architectures. Note also that, the training time of LightGBM for this problem is also greater than that of MTL w/ embedding.

2) COMBINATORY APPROACH - ENSEMBLE AVERAGING OF GBDT AND randECOC ENSEMBLES

As an important outcome of the comparative experiments presented in Section IV-D1, the averaging ensembles tend to achieve the highest accuracy for all the base networks and dataset combinations, benefiting from their increased computational complexity. Under the assumption of an adequate computational resources, we aim further to improve this accuracy by assisting the averaging process with GBDT and randECOC, as explained in IV-C.

The results of these experiments are provided in Table 5. It can be observed that GBDT+averaging approaches outperform the baseline averaging ensemble by the slightest margin, while the randECOC+averaging methods provide a higher performance improvement, ranging from 0.05 up to 2 percentage points, where the highest improvement is observed for the CIFAR-100 dataset.

Although the best accuracies are acquired from randECOC using independent classifiers, MTL based approaches follow closely, revealing better accuracy than GBDTs in all scenarios other than one (Inception-V3 with SVHN), where the difference in performance with the best GBDT framework (XGBoost) is as small as 0.02%. The consistency in the improvement in accuracy not only over the base network, but also the baseline averaging ensemble and the GBDT+averaging ensemble, renders randECOC+averaging as the best performing classifier combination technique in the literature.

[1] except for one out of the nine settings, where a slight drop for the MTL approach was noted.

TABLE 6. 5-Fold cross validation and the combinatory approach on the Plant Village dataset using the Xception base network. The best results obtained in each group are shown in bold.

		Gradient Boosting Ensemble			randECOC Ensemble		
	Base Network	XGBoost	LightGBM	CatBoost	Independent Classifiers	MTL	MTL w/ embedding
Fold-1	99.52%	99.32%	**99.41%**	**99.41%**	**99.71%** ±0.008	99.68% ±0.011	99.68% ±0.009
Fold-2	99.60%	99.53%	99.46%	**99.54%**	**99.73%** ±0.014	99.66% ±0.009	99.68% ±0.011
Fold-3	99.44%	99.36%	99.27%	**99.41%**	**99.70%** ±0.012	99.59% ±0.013	99.61% ±0.008
Fold-4	99.38%	99.36%	**99.41%**	99.41%	**99.67%** ±0.011	99.60% ±0.012	99.64% ±0.014
Fold-5	99.60%	99.40%	99.32%	**99.52%**	**99.71%** ±0.009	99.64% ±0.011	99.68% ±0.008
Average	99.51%	99.40%	99.37%	**99.46%**	**99.70%**	99.63%	99.66%
Combinatory Approach							
	Averaging of 5 base networks	XGBoost	LightGBM	CatBoost	Independent Classifiers	MTL	MTL w/ embedding
Fold 1	99.76%	99.72%	99.74%	99.72%	**99.81%**	99.79%	99.79%

We would like to underline the fact that the GBDT and randECOC frameworks operate on the features extracted by the base networks; hence training the combinatory approach with these frameworks takes little additional time. For instance, training 5 randECOC ensembles on top of the 5 base networks only takes 21 minutes for the CIFAR-10 dataset, while training the 5 base networks takes 3160 minutes. The additional time corresponds to 0.58% overhead.

3) EXPERIMENTS WITH REAL-LIFE PlantVillage DATASET

Experiments on PlantVillage dataset [37] are done using 5-fold cross-validation due to the lack of a designated test set. Due to large computational requirements, the experiments are conducted using only the Xception network, because of its favorable performance-size ratio, and the combinatory approach is applied on only one fold.

The results are shown in Table 6, where it can be observed that while all the performances are very close, the randECOC variants achieve superior accuracy in all folds. Moreover, the combinatory approach of randECOC achieves the state-of-the-art results (99.81%) on this dataset, surpassing Mohanty *et al.*, Too *et al.*, and KC *et al.* who reported %99.34, %99.75, and %98.34 respectively [53]–[55].

V. CONCLUSION

In this paper, we have proposed different design methodologies to address the use of the Error Correcting Output Coding (ECOC) framework as a strategy for constructing deep convolutional neural network ensembles. This is the first study to date, which comprehensively analyses ECOC in relation to the deep learning research, while proposing novel strategies to focus on the accuracy-complexity trade-off.

The current state-of-the-art deep ensemble techniques in the literature are constructed either by averaging the outputs of the multiple realisations of a deep network architecture by randomising / changing some of its constitutional elements, or by employing gradient boosting decision trees (GBDT) on the features extracted from one fully trained network. Despite all its advantages in terms of the performance gain, the increased time complexity the averaging ensembles incur, which is shown to be of the order of days and weeks for problems involving a high number of classes, may make this method computationally infeasible or inefficient for users with limited resources. Even though GBDTs address this inefficiency, they have been shown to be unstable in terms of the improvement they offer over the base networks. In our experiments, we have shown that there exists no GBDT method which provides consistent improvement over the base accuracy for all architectures and datasets.

Addressing the drawbacks of GBDTs, we have proposed and analysed three ECOC-based design techniques, which provide a reliable and stable improvement over the base network performance as well as the performance of GBDT under all settings. Moreover, two of the proposed designs achieve time complexity benefits similar to GBDTs.

The proposed design techniques are based on independent learning, multi-task learning (MTL), and multi-task learning with embedding (MTL w/ embedding). It has been shown that MTL w/ embedding always provides an accuracy equal to or greater than that of MTL, and both methods have a comparable time complexity with those of GBDTs. Independent learning provides the best performance among the ECOC based methods. However, the performance gain over the MTL based methods is marginal and comes with the time complexity trade-off, though this complexity is still much less than that of averaging. Therefore, for problems to be tackled with a limited computational resources, we suggest that employing ECOC methods, the choice of which is to be made by the user depending on the fine-tuned requirements of the problem, is the best strategy; i.e. MTL w/ embedding for fastest training, independent learning for a relatively slower but marginally better performance.

To offer solutions for scenarios where the available resources are not a limiting factor for the user, we have conducted experiments with simple averaging ensembles of GBDT and ECOC frameworks, and shown that the combinatory framework built using any of the ECOC methodologies achieves the best performance, at the expense of negligible additional training time.

In conclusion, the ECOC framework, either alone or in combination with the averaging methodology, appears to provide the most efficient ensemble learning approach. In the

future, the feasibility of end-to-end training of the proposed design strategies using the ECOC framework will be explored for the cases where time and space complexity is not a restriction.

REFERENCES

[1] L. S. Penrose, "The elementary statistics of majority voting," *J. Roy. Stat. Soc.*, vol. 109, no. 1, pp. 53–57, 1946.
[2] L. Breiman, "Bagging predictors," *Mach. Learn.*, vol. 24, no. 2, pp. 123–140, Aug. 1996.
[3] S. Džeroski and B. Ženko, "Is combining classifiers with stacking better than selecting the best one?" *Mach. Learn.*, vol. 54, no. 3, pp. 255–273, Mar. 2004.
[4] L. Breiman, "Random forests," *Mach. Learn.*, vol. 45, no. 1, pp. 5–32, 2001.
[5] E. B. Kong and T. G. Dietterich, "Error-correcting output coding corrects bias and variance," in *Machine Learning Proceedings*. Amsterdam, The Netherlands: Elsevier, 1995, pp. 313–321.
[6] V. Guruswami and A. Sahai, "Multiclass learning, boosting, and error-correcting codes," in *Proc. 12th Annu. Conf. Comput. Learn. Theory (COLT)*, 1999, pp. 145–155.
[7] Y. Taigman, M. Yang, M. Ranzato, and L. Wolf, "DeepFace: Closing the gap to human-level performance in face verification," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2014, pp. 1701–1708.
[8] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 1–9.
[9] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 779–788.
[10] Y. Xiao, J. Wu, Z. Lin, and X. Zhao, "A deep learning-based multi-model ensemble method for cancer prediction," *Comput. Methods Programs Biomed.*, vol. 153, pp. 1–9, Jan. 2018.
[11] N. Gessert, M. Nielsen, M. Shaikh, R. Werner, and A. Schlaefer, "Skin lesion classification using ensembles of multi-resolution EfficientNets with meta data," *MethodsX*, vol. 7, Jan. 2020, Art. no. 100864.
[12] P. Badjatiya, S. Gupta, M. Gupta, and V. Varma, "Deep learning for hate speech detection in tweets," in *Proc. 26th Int. Conf. World Wide Web Companion (WWW) Companion*, 2017, pp. 759–760.
[13] X. Ren, H. Guo, S. Li, S. Wang, and J. Li, "A novel image classification method with CNN-XGBoost model," in *Proc. Int. Workshop Digit. Watermarking*. Cham, Switzerland: Springer, 2017, pp. 378–390.
[14] L. Pang, J. Wang, L. Zhao, C. Wang, and H. Zhan, "A novel protein subcellular localization method with CNN-XGBoost model for Alzheimer's disease," *Frontiers Genet.*, vol. 9, p. 751, Jan. 2019.
[15] Y. Ju, G. Sun, Q. Chen, M. Zhang, H. Zhu, and M. U. Rehman, "A model combining convolutional neural network and LightGBM algorithm for ultra-short-term wind power forecasting," *IEEE Access*, vol. 7, pp. 28309–28318, 2019.
[16] M. Lasseck, "Image-based plant species identification with deep convolutional neural networks," in *Proc. CLEF Working Notes*, Sep. 2017, pp. 1–11.
[17] M. Tan and Q. V. Le, "EfficientNet: Rethinking model scaling for convolutional neural networks," 2019, *arXiv:1905.11946*. [Online]. Available: http://arxiv.org/abs/1905.11946
[18] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Aug. 2016, pp. 785–794.
[19] G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, and T.-Y. Liu, "LightGBM: A highly efficient gradient boosting decision tree," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 3146–3154.
[20] L. Prokhorenkova, G. Gusev, A. Vorobev, A. V. Dorogush, and A. Gulin, "CatBoost: Unbiased boosting with categorical features," in *Proc. Adv. Neural Inf. Process. Syst.*, 2018, pp. 6638–6648.
[21] A. Torres-Barrán, Á. Alonso, and J. R. Dorronsoro, "Regression tree ensembles for wind energy and solar radiation prediction," *Neurocomputing*, vols. 326–327, pp. 151–160, Jan. 2019.
[22] A. Hocquenghem, "Codes correcteurs d'erreurs," *Chiffers*, vol. 2, no. 2, pp. 56–147, 1959.
[23] R. C. Bose and D. K. Ray-Chaudhuri, "On a class of error correcting binary group codes," *Inf. Control*, vol. 3, no. 1, pp. 68–79, Mar. 1960.
[24] T. G. Dietterich and G. Bakiri, "Solving multiclass learning problems via error-correcting output codes," *J. Artif. Intell. Res.*, vol. 2, pp. 263–286, Jan. 1995.
[25] O. Pujol, P. Radeva, and J. Vitria, "Discriminant ECOC: A heuristic method for application dependent design of error correcting output codes," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 28, no. 6, pp. 1007–1012, Jun. 2006.
[26] M. Á. Bautista, S. Escalera, X. Baró, P. Radeva, J. Vitriá, and O. Pujol, "Minimal design of error-correcting output codes," *Pattern Recognit. Lett.*, vol. 33, no. 6, pp. 693–702, Apr. 2012.
[27] G. James and T. Hastie, "The error coding method and PICTs," *J. Comput. Graph. Statist.*, vol. 7, no. 3, pp. 377–387, Sep. 1998.
[28] G. James, "Majority vote classifiers: Theory and applications," Ph.D. dissertation, Dept. Statist., Stanford Univ., Stanford, CA, USA, 1998.
[29] S. Escalera, O. Pujol, and P. Radeva, "On the decoding process in ternary error-correcting output codes," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 32, no. 1, pp. 120–134, Jan. 2010.
[30] L. Xiao-feng, Z. Xue-ying, and D. Ji-kang, "Speech recognition based on support vector machine and error correcting output codes," in *Proc. 1st Int. Conf. Pervasive Comput., Signal Process. Appl.*, Sep. 2010, pp. 336–339.
[31] Q. Ye, J. Liang, and J. Jiao, "Pedestrian detection in video images via error correcting output code classification of manifold subclasses," *IEEE Trans. Intell. Transp. Syst.*, vol. 13, no. 1, pp. 193–202, Mar. 2012.
[32] R. S. Smith and T. Windeatt, "Facial action unit recognition using multi-class classification," *Neurocomputing*, vol. 150, pp. 440–448, Feb. 2015.
[33] S. Gu, Y. Cai, J. Shan, and C. Hou, "Active learning with error-correcting output codes," *Neurocomputing*, vol. 364, pp. 182–191, Oct. 2019.
[34] B. Zhang, B. Tondi, X. Lv, and M. Barni, "Challenging the adversarial robustness of DNNs based on error-correcting output codes," 2020, *arXiv:2003.11855*. [Online]. Available: http://arxiv.org/abs/2003.11855
[35] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," Citeseer, Univ. Toronto, Toronto, ON, Canada, Tech. Rep., 2009.
[36] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng, "Reading digits in natural images with unsupervised feature learning," in *Proc. NIPS Workshop Deep Learn. Unsupervised Feature Learn.*, Granada, Spain, 2011, pp. 12–17.
[37] D. P. Hughes and M. Salathe, "An open access repository of images on plant health to enable the development of mobile disease diagnostics," 2015, *arXiv:1511.08060*. [Online]. Available: http://arxiv.org/abs/1511.08060
[38] B. Efron, "Bootstrap methods: Another look at the jackknife," in *Breakthroughs Statistics*. New York, NY, USA: Springer, 1992, pp. 569–593.
[39] R. E. Schapire, "The boosting approach to machine learning: An overview," in *Nonlinear Estimation Classification*. New York, NY, USA: Springer, 2003, pp. 149–171.
[40] D. H. Wolpert, "Stacked generalization," *Neural Netw.*, vol. 5, no. 2, pp. 241–259, Jan. 1992.
[41] A. C. Lorena and A. C. P. L. F. de Carvalho, "Building binary-tree-based multiclass classifiers using separability measures," *Neurocomputing*, vol. 73, nos. 16–18, pp. 2837–2845, Oct. 2010.
[42] E. Alpaydin, "Learning error-correcting output codes from data," in *Proc. 9th Int. Conf. Artif. Neural Netw., (ICANN)*, 1999, pp. 743–748.
[43] C. Zor, B. Yanikoglu, T. Windeatt, and E. Alpaydin, "FLIP-ECOC: A greedy optimization of the ECOC matrix," in *Proc. 25th Int. Symp. Comput. Inf. Sci. (ISCIS)*. Dordrecht, The Netherlands: Springer, 2010, pp. 149–154.
[44] C. Zor, B. A. Yanikoglu, E. Merdivan, T. Windeatt, J. Kittler, and E. Alpaydin, "BeamECOC: A local search for the optimization of the ECOC matrix," in *Proc. 23rd Int. Conf. Pattern Recognit., (ICPR)*, Cancún, Mexico, Dec. 2016, pp. 198–203.
[45] S. A. Aly and B. Yanikoglu, "Multi-label networks for face attributes classification," in *Proc. IEEE Int. Conf. Multimedia Expo Workshops (ICMEW)*, Jul. 2018, pp. 1–6.
[46] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "MobileNetV2: Inverted residuals and linear bottlenecks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 4510–4520.
[47] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 2818–2826.
[48] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 1251–1258.

[49] J. Hu, L. Shen, and G. Sun, "Squeeze-and-excitation networks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 7132–7141.
[50] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2009, pp. 248–255.
[51] A. Vedaldi and K. Lenc, "MatConvNet: Convolutional neural networks for MATLAB," in *Proc. 23rd ACM Int. Conf. Multimedia*, Oct. 2015, pp. 689–692.
[52] I. E. Kuralenok, Y. Rebryk, R. Solovev, and A. Ermilov, "Factorized MultiClass boosting," 2019, *arXiv:1909.04904*. [Online]. Available: http://arxiv.org/abs/1909.04904
[53] S. P. Mohanty, D. P. Hughes, and M. Salathé, "Using deep learning for image-based plant disease detection," *Frontiers Plant Sci.*, vol. 7, p. 1419, Sep. 2016.
[54] E. C. Too, L. Yujian, S. Njuki, and L. Yingchun, "A comparative study of fine-tuning deep learning models for plant disease identification," *Comput. Electron. Agricult.*, vol. 161, pp. 272–279, Jun. 2019.
[55] K. Kc, Z. Yin, M. Wu, and Z. Wu, "Depthwise separable convolution architectures for plant disease classification," *Comput. Electron. Agricult.*, vol. 165, Oct. 2019, Art. no. 104948.

SARA ATITO ALI AHMED (Member, IEEE) received the B.Sc. degree in computer science from Ain Shams University, Egypt, in 2011, and the joint M.Sc. degree from Nile University, Egypt, and TU Berlin, Germany, in 2014. She is currently pursuing the Ph.D. degree with Sabanci University, thesis on deep learning ensembles for image understanding. In 2013, she was an Intern with the Speech and Sound Group, Sony Deutschland GmbH, Stuttgart, Germany, working on character recognition in natural images. She is working on vehicles detection and tracking in crowded scenes and sensitivity analysis of deep neural networks. She is also a Fellow Researcher with the CVSSP Group, University of Surrey, U.K., working on detection and generation of face morphing.

CEMRE ZOR received the M.Sc. and Ph.D. degrees from the Centre for Vision, Speech and Signal Processing, University of Surrey. She is a Senior Research Associate at the Centre for Medical Image Computing, University College London. Prior to this, she was a Research Associate with the University of Surrey. Her current research interests include multi-modal predictive analysis and anomaly detection in neuroscience data, multiple classifier systems, and the theory of classification.

MUHAMMAD AWAIS received the B.Sc. degree in mathematics and physics from AJK University, in 2001, the B.Sc. degree in computer engineering from UET Taxila, in 2005, the M.Sc. in signal processing and machine intelligence, and the Ph.D. degree in machine learning from the University of Surrey, in 2008 and 2011, respectively. He is currently a Senior Research Fellow with the Centre for Vision, Speech and Signal Processing (CVSSP), University of Surrey. His research interests include machine learning, deep learning, self (un, semi)-supervised learning, NLP, audio-visual analysis, medical image analysis, and computer vision.

BERRIN YANIKOGLU (Senior Member, IEEE) received the double degree in computer science and mathematics from Bogazici University, Turkey, in 1988, and the Ph.D. degree in computer science from Dartmouth College, USA, in 1993. She is currently a Professor in computer science and the Director of the Center of Excellence in Data Analytics (VERIM), Sabanci University, Istanbul, Turkey. She worked at The Rockefeller University, Xerox Imaging Systems, and IBM Almaden Research Center, before joining Sabanci University, in 2000. Her research interests include on image/video understanding, biometric verification and privacy, and handwriting recognition. She is an Editor for the *Turkish Journal of Electrical Engineering and Computer Sciences*.

JOSEF KITTLER (Life Member, IEEE) received the B.A., Ph.D., and D.Sc. degrees from the University of Cambridge, in 1971, 1974, and 1991, respectively.

He is currently a Distinguished Professor in machine intelligence with the Centre for Vision, Speech and Signal Processing, University of Surrey, Guildford, U.K. He published the textbook *Pattern Recognition: A Statistical Approach* and over 700 scientific articles. His publications have been cited more than 68 000 times (Google Scholar). His research interests include biometrics, video and image dataset retrieval, medical image analysis, and cognitive vision. He is currently a member of the K-S Fu Prize Committee of IAPR. He is a Series Editor of Springer Lecture Notes on computer science. He is currently serves on the editorial boards for *Pattern Recognition Letters*, *Pattern Recognition and Artificial Intelligence*, and *Pattern Analysis and Applications*. He is also served as a member for the Editorial Board of IEEE TRANSACTIONS ON PATTERN ANALYSIS and MACHINE INTELLIGENCE, from 1982 to 1985. He served for the Governing Board of the International Association for Pattern Recognition (IAPR) as one of the two British representatives, from 1982 to 2005, and the President of the IAPR, from 1994 to 1996.