DEEP LEARNING ENSEMBLES FOR IMAGE UNDERSTANDING

by
SARA ATITO ALI AHMED

Submitted to the Faculty of Engineering and Natural Sciences
in partial fulfilment of
the requirements for the degree of Doctor of Philosophy

Sabancı University
July 2021

DEEP LEARNING ENSEMBLES FOR IMAGE UNDERSTANDING

Approved by:

Prof. Berrin Yanıkoğlu .....
(Dissertation Supervisor)

Prof. Özgür Gürbüz .....
(Sabanci University)

Assist. Prof. Öznur Taştan .....
(Sabanci University)

Prof. Erchan Aptoula .....
(Gebze Technical University)

Assoc. Prof. Elif Karslıgil .....
(Yıldız Technical University)

Date of Approval: July 09, 2021

SARA ATITO ALI AHMED

Computer Science and Engineering, Ph.D DISSERTATION, July 2021

Dissertation Supervisor: Prof. Berrin Yanıkoğlu

Keywords: Deep Learning, Ensemble Learning, Face Attributes Classification, Multi-label Learning, Multi-task Learning, Error Correcting Output Codes, Skin Lesion Classification, Plant Identification

ABSTRACT

Deep neural networks have enhanced the performance of decision making systems in many applications, including image understanding. Further performance gains can be achieved by using ensemble methods, which are shown to be powerful tools for various classification and regression tasks. This dissertation consists of two parts. The first part is devoted to studying the face attributes classification problem. We introduce several novel approaches for this problem, achieving state-of-art results on CelebA and LFWA datasets: i) we use the multi-task learning (MTL) framework for multiple attributes classification for scalability, where base learners are grouped according to the location of the attribute on the face and share weights. Giving information about the location of an attribute as prior information is shown to speed up the learning process and lead to increased accuracy. ii) we introduce a novel ensemble learning technique within the deep learning model itself (within-network ensemble), showing increased performance at almost the same time complexity of a single model. iii) we propose a new framework called Deep-RankSVM for relative attribute classification (comparing the attribution expression on two photographs) adapting the SVM formulation to deep rank learning.

The second part is devoted to analyzing the suitability of different state-of-art design strategies for constructing ensembles of deep networks. We propose the Error Correcting Output Codes (ECOC) framework as a novel deep learning ensemble method, and show that it can be used with the MTL framework for arbitrary 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, on several datasets. In the rest of the dissertation, we discuss general applications of the proposed ensemble techniques that include skin lesion classification and plant identification.

ACKNOWLEDGEMENTS

First and foremost, I would like to express my gratitude to my supervisor, Prof. Berrin Yanıkoğlu, for her invaluable supervision, motivation, and immense knowledge during the course of my PhD degree. I believe that this work would have never been accomplished without her assistance and dedicated involvement in every step throughout the process. I could not have imagined having a better advisor and mentor. She demonstrated what a brilliant and hard-working scientist can accomplish. Her enthusiasm for the research made a strong impression on me and has inspired me to become an independent researcher.

Besides my advisor, I would like to thank Assoc. Prof. Erchan Aptoula and Dr. Cemre Zor for all the valuable insights and the precious points that they have raised in our discussions.

I would also like to express my special thanks to Dr. Gülşen Demiröz whom I did most of my teaching assistantship with. I really appreciate all the opportunities she has given to me to discover my passion for teaching.

In addition, I want to acknowledge the help provided by Osman Rahmi Fıçıcı for all of the technical support.

I would also like to thank my friends, Melike Gezen, Atia Shafique, Naida Fetic, Sandra Saghir, Ammar Saleem , Nima Zoghipour, Turkhan Khalilov, Wael Aldulaimi and my lab mates Mehmet Yavuz and Mehmet Umut Sen, for all of the cherished time we spent together in the social settings and in the lab.

My appreciation also goes out to my family who offered me a lot of encouragement which were mostly through phone calls every week over the last several years.

Last but not least, a special thanks to my beloved fiancé, Pouya Zoghipour, for all the love and constant support.

Dedicated to my beloved family.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

1. GENERAL INTRODUCTION

Deep learning is one of the most significant achievements in the artificial intelligence domain, leading to significantly enhanced performance in many domains, including image understanding applications (scene classification [1], semantic segmentation [2], and object detection [3]). This dissertation studies deep learning ensembles in the context of image understanding problems. We focus on image classification in general and specifically face attributes classification.

Describing facial attributes such as hair style, gender, and smile, is very beneficial in large scale applications [4] like face recognition and identification [5], face verification [6, 7], and image understanding [8]. However, being able to automatically describe face attributes from images is a challenging task due to the different variations in the images like illuminations, occlusions, poses and background variations. Therefore, attribute classification approaches based on handcrafted representations, as in [7, 9, 10], are prone to failing when presented different variations of face images and in unconstrained backgrounds. Recently, researchers tackle this task using deep learning, which has resulted in huge performance leaps in several domains [11, 12, 13, 14, 15, 16].

The pipeline of face attributes classification consists of three main steps, i.e. face localization, feature extraction, and attributes classification. We mainly focus on visual attributes classification that indicate the presence or absence of a certain semantic attribute. Particularly, we approached the attributes classification task in a Multi-Task Learning (MTL) scenario by grouping attributes based on their localization and sharing weights of each group of attributes. The motivation behind grouping the attributes is the fact that valuable information can be obtained from the correlation of attributes. Besides, grouping attributes not only reduced number of needed classifiers to classify different attributes, but also sharing weights helped reducing overfitting. Furthermore, we speed up the training by directing the attention of the network to the area of interest for a group of attributes employing

pre-defined masks. Moreover, we extended the grouping attributes idea to an end-to-end network where all the attributes are trained at once in a multi-label learning scenario. An extra layer along with a combined objective function are added to the network to capture the relation between the attributes. Additionally, a novel within-network ensemble technique is introduced that reduces the training time compared to ensemble of several models.

Beside binary attributes learning, we also investigated the relative attributes problem where the goal is to learn a function which predicts the relative strengths of a pair of images regarding a given attribute. Specifically, we tackled the relative attributes problem using an end-to-end deep convolutional Siamese network in which the network jointly learns the visual features with the rank SVM objective function.

The employed methodologies for binary and relative visual attributes classification along with extensive experimental results are covered in details in Chapter 2. The contents of this chapter have been published in three conferences. Chapter 2.1 is published in the International Conference of Multimedia & Expo workshops (ICMEW), 2018 [17], Chapter 2.2 is published in the International Conference on Image Analysis and Processing (ICIAP), 2019 [18], and Chapter 2.3 is published in the ICPR International Workshops and Challenges, 2021 [19].

Despite the aforementioned success of deep learning in image understanding, deep neural networks having a high variance as they mostly learn via stochastic training scheme which means that the trained network may produce different predictions when trained with different training configurations. One of the most common ways to reduce variance and improve prediction performance and robustness of deep neural networks is ensemble learning. Classifier ensembles [20, 21] are pattern recognition structures composed of a weighted combination of multiple classification models. 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.

In Chapter 3, we mainly focus on ensembles in deep learning. Several ensemble techniques such as simple/weighted averaging, majority voting [22], error correcting output coding [23], stacking [24], have been widely used in traditional machine learning. However, some of these approaches are impractical/inefficient to deep learning systems due to the computational complexity associated with the training of deep networks. Therefore, most of the state-of-art deep learning ensembles are either formed of simple averaging frameworks [25, 26, 27], or weak decision tree ensembles [28, 29, 30].

Averaging ensembles in deep learning is the process of creating multiple models which are mainly obtained by modifying the various deep learning elements such as the network architectures, and their parameters, data augmentation techniques, etc. One of the main drawbacks of averaging ensembles is the increased time complexity which scales linearly with the addition of each base classifier network. In this work, we propose an efficient deep learning framework based on error correcting output coding (ECOC) to address the shortcomings of time complexity associated with averaging ensemble. ECOC 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. Specifically, we analyse different implementation strategies for random ECOC matrices, by investigating three different design procedures. The first approach is the straight forward approach by training the base classifiers independently according to a given random ECOC matrix. Although all independent tasks can potentially be trained in parallel, this framework might be unattractive under the assumption of limited resources, despite the performance gain. To mitigate this issue, we consider the idea of simultaneous training of the base classifiers by employing multi-task learning for faster training. In addition, we extended the aforementioned multi-task network with embedded error correction layer with weights set from the random ECOC codewords and the output nodes representing the original classes. This error correction 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. Chapter 3 has been published in the IEEE access journal, volume 9, 2021 [31].

In the rest of the dissertation, we discuss general applications of the proposed ensemble techniques. In Chapter 4, we present a deep learning method for skin lesion classification. Specifically, we combine deep convolutional networks with the ECOC framework in a novel way to address the class-imbalanced problem arises in most of the skin lesion datasets. In Chapter 5, we applied deep learning ensembles in plant identification system. Our approach is based on ensembles obtained with different weighted combinations of several deep learning architectures and an ensemble based on deep learning features but uses ECOC as the ensemble. The skin lesion classification approaches (Chapter 4.1 and 4.2) are published in the Signal Processing and Communications Applications Conference (SIU), 2020 [32] and the international conference on Machine Learning, Optimization, and Data Science (LOD), 2020 [33], respectively. The plant identification systems (Chapter 5.1 and 5.2) are published in the Conference and Labs of the Evaluation Forum (CLEF), 2017 [34] and 2018 [35].

2. Attributes Based Deep Learning Approaches

2.1 Multi-Label Networks for Face Attributes Classification

Face attributes classification is drawing attention as a research topic with applications in multiple domains, such as video surveillance and social media analysis. In most attribute classification systems in literature, independent classifiers are trained separately for each attribute. In this work, we propose to train attributes in groups based on their localization (head, eyes, nose, cheek, mouth, shoulder, and general areas) in a multi-task learning scenario to speed up the training process and to prevent overfitting. We have evaluated the idea of using the location knowledge for a particular attribute group to speed up the network training. Attention is drawn to the area of interest by blurring training images outside the region of interest, fine-tuning the system and freezing the earlier layers before continuing training with original images. Several data augmentation techniques are also performed to reduce overfitting. Our approach outperforms the state-of-the-art of the attributes on the public LFWA dataset, with an average improvement of almost 0.7% points. The accuracy ranges from 78% (detecting oval face or shadow on the face) to 97.4% (detecting blond hair) across the attributes.

2.1.1 Introduction

Detecting facial attributes, such as hair style, gender, and smile, is very beneficial in large scale applications [4] like face recognition and identification [5], face verification [6, 7], and image understanding [8]. However, being able to automatically describe

face attributes from images is a challenging task, as real-life images have different illuminations, occlusions, poses and background variations.

Automatic recognition of face attributes became an active research topic, especially with the release of CELEBA and LFWA attribute datasets with more than 200,000 images, each with 40 attribute annotations, by Liu et al. [14].

The general pipeline of face attribute classification can be summarized as follows: (1) Face localization; (2) Feature extraction; (3) Attributes classification. Face localization is outside the scope of this paper, as we work on aligned images. Feature extraction and classification have been addressed separately in the past [9, 7], while newer approaches based on deep learning and especially Convolutional Neural Networks (CNNs) address both problems at once.

In spite of the fact that valuable information can be obtained from the correlation of attributes, most of the state-of-the-art methods are dealing with attributes independently. In this paper, we approached this task in a Multi-Task Learning (MTL) scenario by grouping attributes based on their localization and sharing weights of each group of attributes, also suggested in [36, 37]. Grouping attributes not only reduced number of needed classifiers to classify 40 different attributes, but also sharing weights helped reducing overfitting. We also speed up the training by indicating the area of interest for a group of attributes (e.g. mouth region for smile and wearing lipstick attributes, in a two-stage learning. The main contributions of this paper are as follows:

- Proposing a state-of-the-art approach for face attribute classification, using the Multi-Task Learning framework and various forms of data augmentation in order to reduce overfitting. Our results are evaluated on a well known dataset (LFWA), obtaining an average improvement of almost 0.7% points and maximum relative improvement of 3.77% over the state-of-the-art.
- Suggesting a simple method for passing prior information about the general location of an attribute group, to direct network's attention in order to speed up convergence. We show that the two-stage training (with first blurred images and then original) is both faster and slightly more accurate (Fig. 2.4).

2.1.2 Related Works

Until recent years, facial attributes classification has been addressed with hand-crafted representations, as in [9, 7, 10]. This kind of approaches may fail with

unconstrained background and different variations of face images. More recently, researchers tackle this task using deep learning, which has resulted in huge performance leaps in several domains [16, 37, 14, 13, 11, 12, 15].

In Zhu et al. [13] and Razavian et al. [11], CNNs are used to extract features from landmarks to train independent classifiers for each attribute. This approach requires an accurate landmarks detection.

Liu et al. [14] use two cascaded convolutional neural networks, for face localization (LNet) and attributes prediction (ANet), replacing the last fully connected layer with a support vector machine classifier. Each attribute classifier was trained separately. Similarly in Zhong et al. [16], attribute prediction is accomplished by leveraging different levels of CNNs. Hand and Chellapa's work [37] is the most similar to ours: they divide the attributes into nine groups and train a CNN consisting of three convolutional sub-networks and two multi-layer perceptrons. The first two convolutional sub-networks are shared for all of the classifiers (representing earlier and shared features) and the rest of the network is independent for each group. They also compare their results to the results of classifiers trained independently for each attribute and show the advantage of grouping attributes together.

2.1.3 Method

Most of the existing work on face attributes classification ignores the relationship between different facial attributes, and trains individual classifiers for each attribute separately. In this work, we propose to train attributes in groups based on their localization (head, eyes, nose, cheeks, mouth, shoulder, and general areas) in a multi-task learning scenario, to speed up the training process and to prevent overfitting. The area of interest for a particular attribute group is indicated by blurring the image outside the attribute group region, based on the mean image of the training set. In our case, 40 different attributes are considered and divided into 7 groups (Table 2.1).

2.1.3.1 Network Architecture and Training

Training a large deep learning network from scratch is time consuming and needs tremendous amount of training data. Therefore, our approach is based on fine-tuning a pre-trained model, namely the VGG19 network [38] which is the winning

architecture of the ImageNet Large-Scale Visual Recognition Challenge (ILSVRC) 2014. VGG19 is trained on a dataset with 1.2 million hand-labeled images of 1,000 different object classes. Its architecture involves 16 convolution layers, five pooling layers and three fully-connected layers.

As we consider the problem as a multi-task learning problem, the output layer is changed to represent the labels in each attribute group and the loss function is replaced with a multi-label sigmoid Loss. For a single image I with A attributes, the cross-entropy error is denoted as shown in Equation 2.1:

$$E(I) = \sum_{a=1}^{A} -y_I[a] \times \hat{y}_I[a] + log(1 + exp(\hat{y}_I[a])) \tag{2.1}$$

where $y_I[a]$ and $\hat{y}_I[a]$ are the target and output of image I indexed by attribute a, respectively.

Table 2.1 Grouping attributes based on their relative location.

Group	Attributes
Head	Black Hair, Blond Hair, Brown Hair, Gray Hair, Bald, Bangs, Straight Hair, Wavy Hair, Receding Hairline, Hat
Eyes	Arched Eyebrows, Narrow Eyes, Bushy Eyebrows, Bags Under Eyes, Eyeglasses
Nose	Big Nose, Pointy Nose
Cheek	5 O-clock Shadow, Rosy Cheeks, Goatee, High Cheekbones, No Beard, Sideburns
Mouth	Big Lips, Smiling, Mustache, Wearing Lipstick, Mouth Slightly Open
Shoulder	Double Chin, Wearing Necklace, Wearing Necktie
General	Attractive, Blurry, Chubby, Young, Male, Pale Skin, Oval Face, Heavy Makeup, Earrings

Multi-Task learning has already shown a significant success in different applications like face detection, facial landmarks annotation, pose estimation, and traffic flow prediction [39, 40, 41, 42]. MTL is mainly applied by sharing all of the hidden layers between the given tasks but with different output layer for each task. As shown in [43], sharing weights for multiple tasks acts as a regularizer that help reducing the risk of overfitting. Intuitively, the model is forced to learn a general representation that captures all of the specified tasks which less the chance of overfitting.

We used the VGGNet models provided in the CAFFE deep learning framework [44]. Throughout this work, we set the batch size equal to 20 with iteration size equal to 2 and the initial learning rate as 10^{-3} with a total of $1K$ iterations for stage 1 and $10K$ iterations for stage 2.

In order to speed up the training and concentrate the feature extraction process into a local region, the training process of each group of attributes is completed in two stages: (1) directing the attention of the network to the area of interest by first training with blurred images outside the area of interest (Sec. 3.2); and (2) and freezing early layer weights and fine-tuning the system using the original dataset (Sec. 3.3).

2.1.3.2 Stage 1: Directing Attention

Training a huge convolutional neural network with a small dataset, especially if ground-truth labels are noisy, requires thousands of iterations to obtain a good representation from the region of interest (ROI). Automatic attention mechanisms have attracted interest in recent years, with the goal of focusing on a small part of the input or attending to past input in a recurrent network [45]. Our goal is simply to direct attention by indicating a small amount of prior information to the network, in order to speed up the convergence. We indicate the location information of a group of attributes to the network by blurring the images outside the ROI, so as to extract most of the features within the desired region. The early weights learned in this stage are then fixed in the next stage.

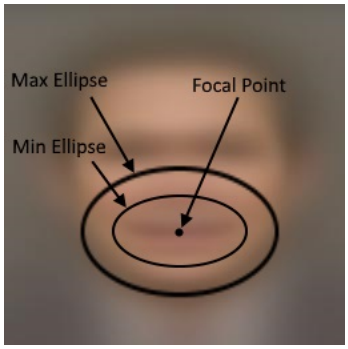
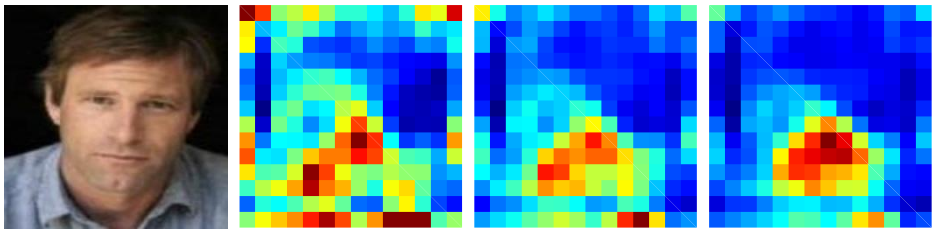


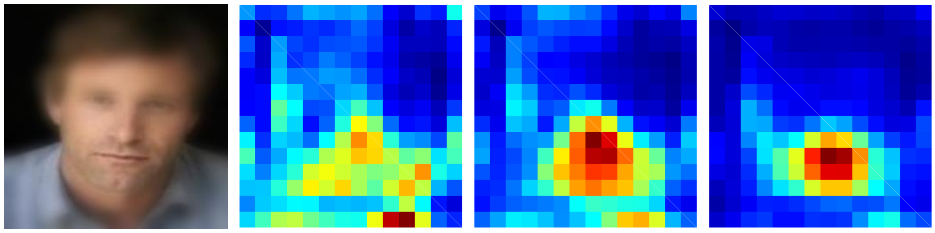
Figure 2.1 Stage 1 for the mouth region: The region outside the ROI is blurred, as defined by the min and max ellipses whose center is detected on the mean training image.

Training images are pre-processed by convolution with an elliptical 2D Gaussian kernel centered on the region of interest, outside the ROI itself, as shown in Figures 2.1.



Original Image 1000 iterations 3000 iterations 5000 iterations

(a) Extracted features using original training images.



Apply Attention 100 iterations 500 iterations 1000 iterations

(b) Extracted features using attention mechanism.

Figure 2.2 Comparison of training the network (a) directly with original training images or (b) by directing attention with blurred images.

The center and the size around the ROI are defined based on the mean image of the dataset. Furthermore, dataset augmentation is also achieved by changing the strength of blur and the size of the ellipse between pre-defined minimum and maximum, as shown in Figure 2.1.

The system input is an image resized to 256×256 and blurred as described above. Then, it undergoes internal data augmentation and gets cropped to 224×224, according to the input layer size of VGG19. The pre-trained VGG19 network is then fine-tuned using the blurred images for 1,000 iterations.

Figure 2.2 shows the summation of the last convolutional layer outputs after different number of iterations by training the network with original images directly (Figure 2.2a) and training the network with pre-processed images by focusing on the region of interest (Figure 2.2b). Neural activations show that the focus of the network is tuned mostly to the region of interest by the end of Stage 1.

In the second stage, we freeze the early layer weights from this stage and fine-tune the rest of the network using original images. In Section 2.1.4 we compare this approach to fine-tuning with original or blurred images in one stage. Our results show that the network learns much faster in our case, as well as having a slightly higher accuracy.

2.1.3.3 Stage 2: Fine Tuning

In this stage, the VGG19 network is fine-tuned by continuing the back-propagation starting from the trained model coming from Stage 1, but by freezing the weights of low-level portion of the network (10 convolutional layers) and using the original images. The learning rate of the rest convolution layers are reduced by factor of 10 to keep learning but sustaining the extracted features from stage 1. Thus, the features that lie outside of the region of interest but might be helpful in classifying the current group of attributes (e.g. eye features being used in smile detection) can be considered.

For data augmentation, we used both internal and external augmentation. For external augmentation, all augmented data are generated before training where several augmentation techniques are used as shown in Section 2.1.4.2. For internal augmentation, each input image is augmented by random cropping and random horizontal flipping, provided optionally in the CAFFE framework [44].

2.1.4 Experiments

2.1.4.1 Dataset

The LFW [46] dataset is used to assess our proposed method. Originally, the dataset is constructed for face identification and verification, while recently, it is annotated with 40 different binary attributes [14]. The annotated dataset (LFWA) is publicly available where it contains 13,143 images of 5,749 different identities. The dataset has a designated training set portion of 6,263 images, while the rest is reserved for testing. LFWA is one of the challenging datasets with large variations in pose, contrast, illumination and image quality.

2.1.4.2 Data Augmentation

In deep learning, data augmentation plays an important role in avoiding overfitting, specially with smaller datasets. Recently, several advanced methods for face data augmentation have been developed. In this paper, simple but effective data augmentation techniques are used: (1) Rotation: training images are rotated using a

random rotation angle between [-5, +5] around the origin. (2) Scaling: images are scaled up and down with a random scale factor up to a quarter of the image size. (3) Contrast: by converting the color space of the images from RGB to HSV and randomly multiplying the S and V channels with a factor range between [0.5, 1.5]. In addition, blurring with two different filter size (3x3 and 5x5) and histogram equalization are performed. Furthermore, some techniques are applied in combination (e.g. rotation and scaling, or rotation and blurring).

We also add another type of augmentation by superimposing accessories such as glasses, hats, and wigs, on the training images. For this, the annotation of eyes locations are used to properly scale and rotate the added accessory. Random samples from the embedded items and generated augmented data are shown on Figure 2.3. In total, we generated 22 images per training sample, which corresponds to expanding our training set to 137,786 samples.

Figure 2.3 Data augmentation with accessories.

2.1.4.3 Results and Evaluation

We compare our work to the results obtained by three state-of-the-art methods, along with the baseline of choosing the most frequent label for each attribute. The performance comparison reported in Table 2.2 shows that our average accuracy compared to the best system (MCNN-AUX [37]) is almost 0.7% higher and outperforms it for 33 of 40 of the attributes. The state-of-art on this dataset has shown a relative increase of 2.46 on average in more than two years.

Considering the results, we see that both our approach and the MCNN-AUX ap-

proach performs better compared to each attribute being trained individually. Thus our results confirm that grouping attributes in a MTL framework is useful.

As for the small but consistent improvements over the state-of-the-art, we believe that there are two reasons: First, we used several data augmentation techniques, whereas the augmentation is done by only jittering the original dataset in [37]. Second, [37] uses a small network that consists of three convolutional stages and two hidden layers and shares weights among different attributes. We believe that using a larger network and sharing the weights only within a regional group allows for a more powerful network, which is then constrained by way of data augmentation to reduce overfitting.

Finally, in order to see the benefits of directing the attention and the two-stage training, we trained 3 systems: System1) applying only the second stage, which corresponds to training in a MTL scenario using the original images without directing attention; System2) using blurring in both stages rather than using original images in Stage 2; and System3) the proposed method.

As can be seen in Figure 2.4, the error drops fastest in the proposed scheme where the system is given a little information about the rough feature location (proposed approach is better than System1), but only enough to direct the attention (proposed approach is better than System2).

Training with original images eventually catches up and even surpasses training with blurred images and also comes close to the proposed method. This is in fact expected, since blurring loses some information.

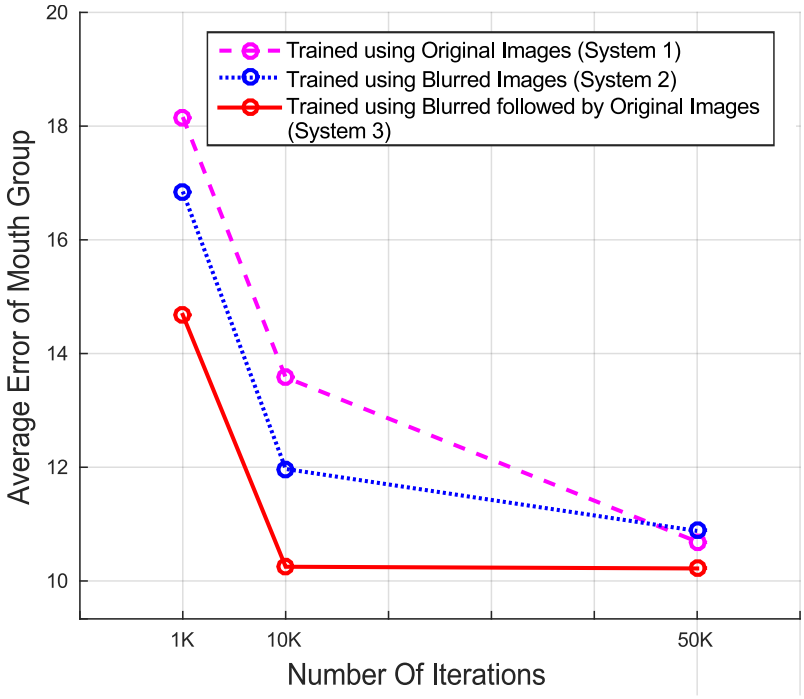


Figure 2.4 Average error of mouth group for 3 systems.

Table 2.2 State-of-the-art accuracies compared with the results obtained in this work. Bold figures indicate the best results.

#	Attribute	Baseline	[14]	[16]	[47]	Independent [37]	MCNN-AUX [37]	Ours
				Head				
1	Black Hair	87.63	90	91	83	91.84	92.63	**92.79**
2	Blond Hair	95.74	97	97	92	97.23	**97.41**	**97.41**
3	Brown Hair	64.56	77	76	97	80.84	80.85	**81.09**
4	Gray Hair	84.25	84	87	**89**	88.98	88.93	88.92
5	Bald	89.37	88	91	**93**	91.51	91.94	92.09
6	Bangs	83.59	88	91	77	90.47	90.08	**91.05**
7	Straight Hair	64.44	76	77	79	81.54	78.53	**82.30**
8	Wavy Hair	55.49	76	77	**94**	81.58	81.61	81.89
9	Reced. Hairline	59.84	85	86	85	86.00	86.26	**86.89**
10	Wear. Hat	85.52	88	90	**92**	89.79	90.07	91.50
				Eyes				
11	Arch. Eyebrows	74.88	82	83	**86**	81.40	81.78	84.01
12	Narrow Eyes	65.50	81	81	82	82.48	82.86	**83.26**
13	Bushy Eyebrows	53.70	82	83	82	84.79	84.97	**85.94**
14	Bags Under Eyes	58.29	83	83	**92**	83.24	83.48	83.01
15	Eyeglasses	81.99	**95**	91	86	92.15	91.30	92.54
				Nose				
16	Big Nose	68.59	81	83	80	84.43	**84.98**	84.80
17	Pointy Nose	71.10	80	83	84	**84.41**	84.14	84.40
				Mouth				
18	Big Lips	62.86	75	78	81	79.06	79.24	**82.24**
19	Smiling	60.50	91	90	92	**92.22**	91.83	92.14
20	Mustache	86.62	92	94	**95**	93.69	93.43	94.14
21	Wear. Lipstick	85.53	95	95	93	94.68	**95.04**	94.46
22	Mouth S. O.	58.70	82	81	**86**	82.41	83.51	85.75
				Cheek				
23	5 O-clock Shadow	58.64	84	77	**80**	77.39	77.06	78.01
24	Rosy Cheeks	79.65	78	82	86	**89.46**	87.92	88.90
25	Goatee	74.68	78	83	**88**	83.34	82.97	82.50
26	H. Cheekbones	67.74	88	88	**89**	88.02	88.38	88.49
27	No Beard	70.05	79	80	81	81.45	82.15	**83.39**
28	Sideburns	68.72	77	82	80	81.70	83.13	**83.49**
				Shoulder				
29	Double Chin	62.44	78	80	**92**	82.00	81.52	81.92
30	Wear. Necklace	80.49	88	90	**91**	89.98	89.94	90.77
31	Wear. Necktie	64.09	79	81	81	80.34	80.66	**81.19**
				General				
32	Attractive	62.87	83	79	**84**	80.20	80.31	80.96
33	Blurry	84.02	74	88	75	86.71	85.23	**86.82**
34	Chubby	63.92	73	75	**78**	75.85	76.86	76.93
35	Young	79.60	86	86	**87**	85.11	85.84	86.06
36	Male	78.77	94	94	93	93.27	94.02	**94.20**
37	Pale Skin	52.09	84	73	91	94.31	93.32	**94.38**
38	Oval Face	51.49	74	75	75	77.06	77.39	**78.01**
39	Heavy Makeup	89.20	95	95	95	95.63	**95.85**	95.47
40	Wear. Earrings	86.86	94	95	80	94.73	94.95	**95.04**
	Average	71.85	83.85	84.78	86.15	86.28	86.31	**86.98**

2.1.5 Conclusion

We presented a multi-task framework for face attribute classification based on feature locality. The grouping of the attributes reduces overfitting, in addition to speeding up the learning process. We also show that by using a little amount of domain knowledge about attributes' locality on the face, the network learns much faster and even slightly increases accuracy. With the use of several data augmentation techniques, the system obtains state-of-art results.

2.1.6 Acknowledgements

We gratefully acknowledge NVIDIA Corporation with the donation of the Titan X Pascal GPU used in this research.

2.2 Within-Network Ensemble for Face Attributes Classification

Face attributes classification is drawing attention as a research topic with applications in multiple domains, such as video surveillance and social media analysis. In this work, we propose to train attributes in groups based on their localization (head, eyes, nose, cheek, mouth, shoulder, and general areas) in an end-to-end framework considering the correlations between the different attributes. Furthermore, a novel ensemble learning technique is introduced within the network itself that reduces the time of training compared to ensemble of several models. Our approach outperforms the state-of-the-art of the attributes with an average improvement of almost 0.60% and 0.48% points, on the public CELEBA and LFWA datasets, respectively.

2.2.1 Introduction

Attribute classifiers have been drawing attention in zero-shot or few-shot learning problems where classes share attributes among them and can thus be recognized with zero or a few samples. Face attribute in particular has been a focus [14, 48, 49, 50, 37], as describing facial attributes has useful applications such as attribute-based search. Previously, work on face attribute classification approaches were based on handcrafted representations, as in [7, 9, 10]. This kind of approaches are prone to failing when presented different variations of face images and in unconstrained backgrounds. Recently, researchers tackle this task using deep learning, which has resulted in huge performance leaps in several domains [11, 12, 13, 14, 15, 16]. Liu et al. [14] use two cascaded convolutional neural networks (CNNs), for face localization (LNet) and attributes prediction (ANet). Each attribute classifier is trained independently where the last fully connected layer is replaced with a support vector machine classifier. Similarly in Zhong et al. [16], attribute prediction is accomplished by leveraging different levels of CNNs.

Lately, the task is shifted to be a multi-task learning (MTL) problem by training attributes in groups, mainly to speed up the training process and reduce overfitting. Yet, only few works address the relationship between different facial attributes [37, 17, 50]. Hand and Chellapa's work divides the attributes into nine groups and train a CNN consisting of three convolutional sub-networks and two multi-layer perceptrons [37]. The first two convolutional sub-networks are shared for all of the classifiers and the rest of the network is independent for each group. They also compare their results to the results of classifiers trained independently for each attribute and show

the advantage of grouping attributes together. Atito and Yanikoglu use the multi-task learning paradigm, where attributes that are grouped based on their location, share separate layers [17]. Learning is done in two-stages: first by directing the attention of each network to the area of interest and then fine-tuning the networks. In Han et al. [50], attributes are grouped into ordinal vs. nominal attributes, where nominal attributes usually have two or more classes and there is no intrinsic ordering among the categories, like race and gender. The attributes are jointly estimated by training a convolutional neural network that consists of some shared layers among all the attributes and category-specific layers for heterogeneous attributes.

In this work, we propose an end-to-end network where all of the attributes are trained at once in a multi-label learning scenario. An extra layer along with a combined objective function are added to the network to capture the relation between the attributes. Furthermore, a novel ensemble technique is introduced.

The main contributions are summarized as follows. (1) We use an end-to-end deep learning framework for face attribute classification, capturing the correlation among attributes with an extra layer that is trained at the same time with the first one. (2) We propose a novel within-network ensemble technique. (3) We obtain state-of-the-art results on both the CELEBA and LFWA datasets.

2.2.2 Proposed Approach

In this paper, we approached the face attributes classification problem in a multi-label/multi-task fashion using an end-to-end framework. In Sec. 2.2.2.1, we trained our base system in a multi-label fashion by sharing the network layers among all of the attributes. While in Sec. 2.2.2.2, we introduced groups and attributes specific layers for distinct feature extraction. In Sec. 2.2.2.3, an extra layer is embedded to the architecture to capture the relation between different attributes. Finally, in Sec. 2.2.2.4, a novel ensemble approach within the architecture itself is introduced.

Training a large deep learning network from scratch is time consuming and needs tremendous amount of training data. Therefore, all of our proposed architectures are based on fine-tuning a pre-trained model, namely the ResNet-50 network [51] which is the first place winner of the (ILSVRC) 2015 classification competition with top-5 error rate of 3.57%, trained on a dataset with 1.2 million hand-labeled images of 1,000 different object classes.

2.2.2.1 Base System

Multi-Task learning has already shown a significant success in different applications like face detection, facial landmarks annotation, pose estimation, and traffic flow prediction [39, 40, 41, 42].

In this work, we use MTL such that all the attributes are trained at once, using the same shared layers. To match the output of ResNet-50 network with our task, the output layer is replaced with 40 output units (one for each attribute) and use the cross-entropy loss function to measure the discrepancy between the expected and actual attribute values.

The multi-task approach not only saves on the training time, but the shared network is also more robust to overfitting, according to our experimental results. Intuitively, the model is forced to learn a general representation that captures all of the specified tasks which less the chance of overfitting. Similar findings are also reported in [43] and attributed to the regularization effect obtained by sharing weights for multiple tasks.

Table 2.3 Grouping attributes based on their relative location.

Group	Attributes
Head	Black Hair, Blond Hair, Brown Hair, Gray Hair, Bald, Bangs, Straight Hair, Wavy Hair, Receding Hairline, Hat
Eyes	Arched Eyebrows, Narrow Eyes, Bushy Eyebrows, Bags Under Eyes, Eyeglasses
Nose	Big Nose, Pointy Nose
Cheek	5 O-clock Shadow, Rosy Cheeks, Goatee, High Cheekbones, No Beard, Sideburns
Mouth	Big Lips, Smiling, Mustache, Wearing Lipstick, Mouth Slightly Open
Shoulder	Double Chin, Wearing Necklace, Wearing Necktie
General	Attractive, Blurry, Chubby, Young, Male, Pale Skin, Oval Face, Heavy Makeup, Earrings

2.2.2.2 Multi-Task Learning with Attribute Grouping

When all the layers are shared in a simple multi-task learning approach, the resulting network may be overly constrained. Therefore, we added a residual block for each group of attributes, after the last residual network block (res5b), as well as few layers for each attribute. This architecture is shown in the dashed part of Fig. 2.5.

For grouping, the 40 attributes defined for the CELEBA and LFWA datasets are divided into 7 groups based on their localization (head, eyes, nose, cheeks, mouth, shoulder, and general areas) as shown in Table 2.3.

In the rest of the paper, we discuss our improvement to the multi-task learning network described thus far.

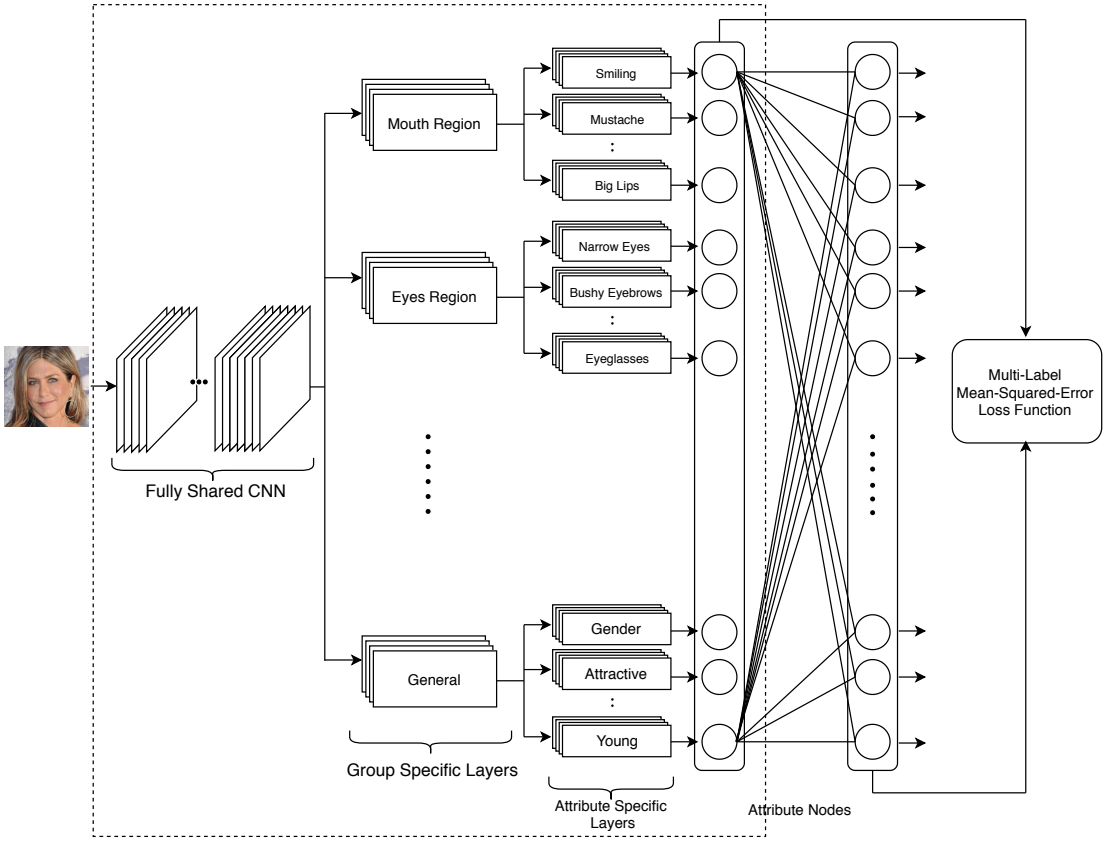


Figure 2.5 End-to-end architecture for face attributes classification.

2.2.2.3 End-to-End Network

Neither the basic, nor the multi-task architectures so far take into account the correlations among attributes.

In previous work, correlations among facial attributes are learned and exploited by using a separate network or learning phase. In this work we add another fully connected layer with 40 output nodes to the network described in Section 2.2.2.2, for simplicity and end-to-end training. The resulting architecture is shown in Fig. 2.5, where the last layer aims to pick the most suitable predictions based on the predictions in the previous layer, by learning the correlations between the attributes.

The multi-label mean-squared-error loss used in this network consists of two terms, one for each of the last two layers. Specifically, for a given input image and A attributes, the loss function is denoted as shown in Equation 2.2, where $\hat{y}_1[a]$ and $\hat{y}_2[a]$ denote the output for attribute a, in the last two layers:

$$loss = \sum_{a=1}^{A} (y[a] - \hat{y}_1[a])^2 + (y[a] - \hat{y}_2[a])^2 \tag{2.2}$$

In this architecture, mean-squared-error loss is used instead of cross-entropy loss, with target values of $\{-1, 1\}$, since we aim to capture attribute correlations with the last layer weights.

2.2.2.4 Within-Network Ensemble

Ensemble approaches are very important in reducing over-fitting and they are used more and more to improving the performance of deep learning systems. However, forming ensembles from deep learning systems is very costly, as training often takes long hours or days.

To reduce the time to build the base classifiers forming the ensemble and inspired by the improved results with the end-to-end architecture with two output layers, we trained an ensemble all at once, within a single network.

The architecture illustrated in Fig. 2.6 shows the main idea behind our approach. Assuming that we have a classification/regression task with N outputs (here the 40 binary attribute nodes), we branch a fully connected layer with N output nodes after every several layers and include their error in the global loss function. During testing, the outputs of these branches are treated as separate base classifier outputs and averaged to obtain the final output.

In this work, we have constructed the ensemble with 5 such branches, each with 40 output nodes. The training of the network for one epoch on the LFWA dataset took approximately 18 minutes, compared to 16 minutes with the end-to-end network.

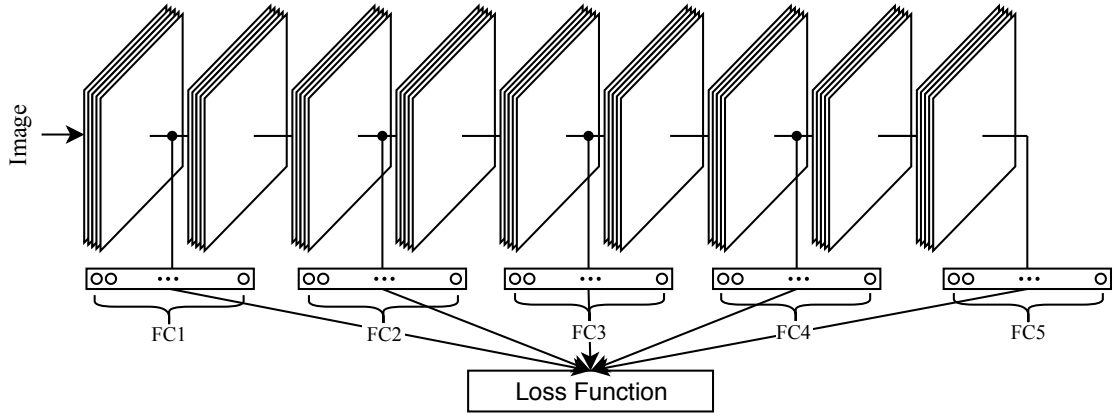


Figure 2.6 A basic architecture of within-network ensemble approach, with 5 output layers.

Notice that the base classifiers formed in this fashion use progressively more complex features and the training is much faster compared to training several separate network as base classifiers. On the other hand, while these base classifiers are not independent from each other, they show complementary behaviour, based on our experimental findings. More implementation details are discussed in Sec. 2.2.3.3.

2.2.3 Experimental Evaluation

We evaluated the effectiveness of our approach using the widely used CELEBA and LFWA datasets, described in Section 2.2.3.1. Data augmentation techniques used while training are presented in Section 2.2.3.2. In Section 2.2.3.3, the network and implementation details are explained. Finally, in Section 2.2.3.4, the performance of our proposed method is evaluated along with a comparison with several state-of-the-art techniques.

2.2.3.1 Datasets

Our experiments are conducted on two well-known datasets for face attributes classification to assess our proposed method, CELEBA and LFWA [14].

CELEBA [14] consists of $202,599$ images of $10,177$ different celebrity faces identities. The first $8k$ identities are set for training (in total around 160k images), while the remaining images are used for validation and testing (around 20k images each). The dataset provides 5 landmark locations (both eyes, nose,

and mouth corners), along with ground-truth for 40 binary attributes for each image.

LFWA [14] is originally constructed for face identification and verification [46], but recently, it is annotated with the same 40 binary attributes. The annotated dataset contains 13,143 images of 5,749 different identities. The dataset has a designated training set portion of 6,263 images, while the rest is reserved for testing. LFWA is one of the challenging datasets with large variations in pose, contrast, illumination and image quality.

2.2.3.2 Data Augmentation

Deep networks typically have large number of free parameters on the order of several millions, which makes the networks prone to overfitting. One way to combat overfitting is to use data augmentation. Recently, several advanced methods for face data augmentation have been developed and automated as in [52].

In this work, we want to show the effectiveness of our stand-alone architecture without using sophisticated data augmentation or pre-processing techniques. Therefore, we only use the following simple, but effective data augmentation techniques: (1) Rotation: training images are rotated using a random rotation angle between [-5, +5] around the origin. (2) Scaling: images are scaled up and down with a random scale factor up to a quarter of the image size. (3) Contrast: by converting the color space of the images from RGB to HSV and randomly multiplying the S and V channels with a factor range between [0.5, 1.5]. In addition, blurring with two different filter size (3x3 and 5x5) and histogram equalization are performed.

At every iteration, we randomly decide whether to apply a transformation to the input image and then pick its parameter randomly. Thus, an input image may undergo a combination of multiple transformations, during one presentation.

2.2.3.3 Network Details and Implementation

As mentioned in Section 2.2.2.3, ResNet-50 is used as our base model in this work, chosen due to its relatively small size and good performance.

All of the layers of ResNet-50 are shared among all of the attributes, up until the last residual block, namely res5b. Then, seven forks are branched from the res5b

layer, one for each group of attributes. Each group's shared layers are similar to the layers in the last residual block of ResNet-50, which are as following: a dropout layer followed by a three consecutive blocks of convolutional layer, batch normalization, scaling and ReLU layer.

After every group block, several forks are branched, one for each attribute: a dropout layer, pool layer, followed by a fully connected layer with one unit. The output coming from all of the branches are then concatenated to form a vector of 40 units and a hyperbolic tangent ($tanh$) activation layer is applied after this layer. Finally, a fully connected layer with 40 units is added at the end, followed by $tanh$ activation layer, to learn the correlations among attributes.

For the within-network ensemble, 5 base classifiers are branched after the res2c, res3c, res4a, res4d and res5a layers of the network. The whole network is trained at once, with 7 terms in the loss function (5 coming from the extra branched layers and 2 from the last two fully connected layers).

The implementation is done using the ResNet-50 models provided in the Matlab deep learning toolbox. Throughout this work, we set the batch size equal to 32 and the initial learning rate as 10^{-3} with a total of 20 epochs with stochastic gradient descent for parameters optimization.

The training of the three models effectively took the same amount of time. Specifically, training ResNet-50 model using LFWA dataset for one epoch was performed in 15.52 minutes with the multi-task learning network, 16.02 minutes with the end-to-end network and 18.28 minutes with the within-network-ensemble approach.

2.2.3.4 Results and Evaluation

A comparison between our proposed methods that are described in Sec. 2.2.2, is shown using the LFWA dataset in Fig. 2.8. We have obtained an average accuracy of 85.15% using the base system approach; 85.66% with the multi-task network using attribute grouping; 85.92% after embedding an extra layer to capture the relation between the attributes; and finally 86.63% using our novel within-network ensemble technique. Our approach outperforms the state-of-the-art results on LFWA ([50]) by 0.48%.

In Fig. 2.7, our within-network ensemble approach is compared with the state-of-the-art accuracies obtained on the larger CELEBA dataset. We obtained an average accuracy of 93.20% that surpasses the state-of-the-art obtained in [50], by 0.60%.

Note that improvements are small due partly to the already high accuracy rates for this problem and the fact that some of the binary attributes are in fact continuous attributes (e.g. smile). By visualizing the learned weights of the last hidden layer (Fig. 2.9), we found that the relationship between attributes are nicely captured. For instance, the learned weights show a high negative correlation between "No Beard" attribute and "Mustache", "Goatee", and "Side Burns" attributes. Contrarily, there is a high positive correlation between "Heavy Makeup" attribute and "Wearing Lipstick", "Rosy Cheeks", and "No Beard" attributes.

State-of-art results on the CELEBA dataset and those obtained with the within-network ensemble are shown in Table 2.4.

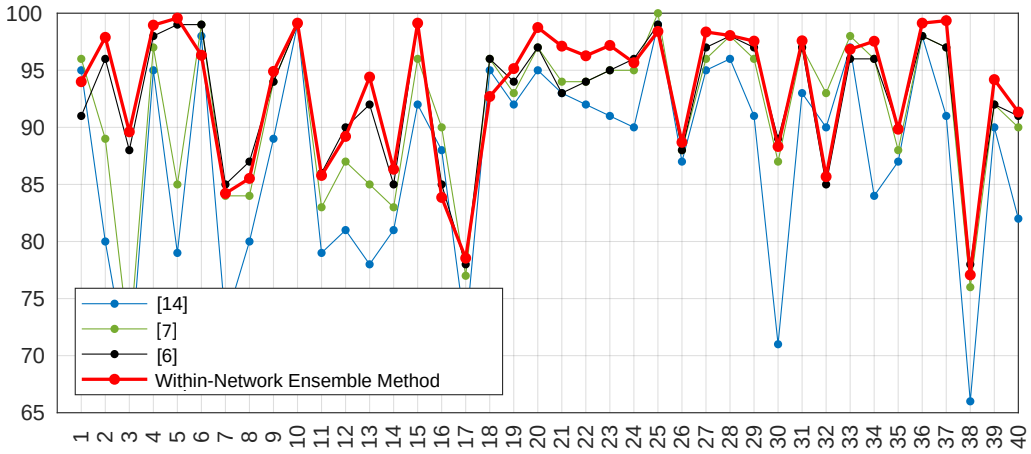


Figure 2.7 State-of-the-art accuracies on CELEBA dataset compared with our proposed approach. Best viewed in color.

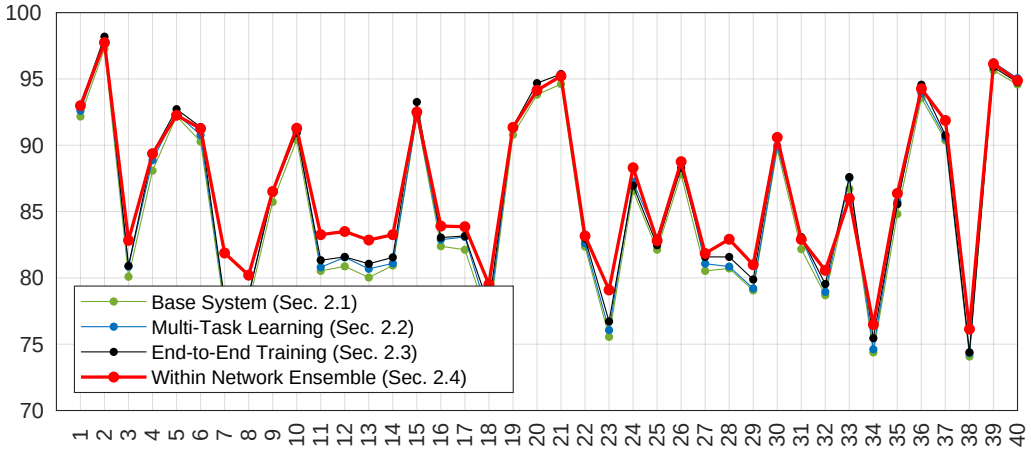


Figure 2.8 Obtained accuracies on LFWA dataset from the increasingly complex networks described in Sec. 2.2.2. Best viewed in color.

Table 2.4 State-of-the-art accuracies on CELEBA dataset compared with the results obtained in this work, using the within-network ensemble. Bold figures indicate the best results.

#	Attribute	Baseline	[14]	[37]	[50]	This Work
			Head Group			
1	Black Hair	72.84%	95%	**96%**	91%	94.00%
2	Blond Hair	86.67%	80%	89%	96%	**97.89%**
3	Brown Hair	82.03%	68%	71%	88%	**89.61%**
4	Gray Hair	96.81%	95%	97%	98%	**98.96%**
5	Bald	97.88%	79%	85%	99%	**99.57%**
6	Bangs	84.43%	98%	**99%**	**99%**	96.32%
7	Straight Hair	79.01%	73%	84%	**85%**	84.21%
8	Wavy Hair	63.60%	80%	84%	**87%**	85.53%
9	Receding Hairline	91.51%	89%	94%	94%	**94.90%**
10	Wearing Hat	95.80%	99%	99%	99%	**99.13%**
			Eyes Group			
11	Arched Eyebrows	71.56%	79%	83%	**86%**	85.79%
12	Narrow Eyes	85.13%	81%	87%	**90%**	89.21%
13	Bushy Eyebrows	87.05%	78%	85%	92%	**94.41%**
14	Bags Under Eyes	79.74%	81%	83%	85%	**86.33%**
15	Eyeglasses	93.54%	92%	96%	99%	**99.13%**
			Nose Group			
16	Big Nose	78.80%	88%	**90%**	85%	83.86%
17	Pointy Nose	71.43%	72%	77%	78%	**78.54%**
			Mouth Group			
18	Big Lips	67.30%	95%	**96%**	**96%**	92.70%
19	Smiling	49.97%	92%	93%	94%	**95.15%**
20	Mustache	96.13%	95%	97%	97%	**98.75%**
21	Wearing Lipstick	47.81%	93%	94%	93%	**97.11%**
22	Mouth Slightly ...	50.49%	92%	94%	94%	**96.27%**
			Cheek Group			
23	5 o'Clock Shadow	90.01%	91%	95%	95%	**97.18%**
24	Rosy Cheeks	92.83%	90%	95%	**96%**	95.66%
25	Goatee	95.42%	99%	**100%**	99%	98.41%
26	High Cheekbones	51.82%	87%	88%	88%	**88.69%**
27	No Beard	14.63%	95%	96%	97%	**98.36%**
28	Sideburns	95.36%	96%	98%	98%	**98.05%**
			Shoulder Group			
29	Double Chin	95.43%	91%	96%	97%	**97.56%**
30	Wearing Necklace	86.21%	71%	87%	**89%**	88.32%
31	Wearing Necktie	92.99%	93%	97%	97%	**97.58%**
			General			
32	Attractive	50.42%	90%	**93%**	85%	85.68%
33	Blurry	94.94%	97%	**98%**	96%	96.84%
34	Chubby	94.70%	84%	96%	96%	**97.54%**
35	Young	24.29%	87%	88%	**90%**	89.84%
36	Male	61.35%	98%	98%	98%	**99.13%**
37	Pale Skin	95.79%	91%	97%	97%	**99.35%**
38	Oval Face	70.44%	66%	76%	**78%**	77.07%
39	Heavy Makeup	59.50%	90%	92%	92%	**94.19%**
40	Wearing Earrings	79.34%	82%	90%	91%	**91.34%**
	Average	76.87%	87.30%	91.32%	92.60%	**93.20%**

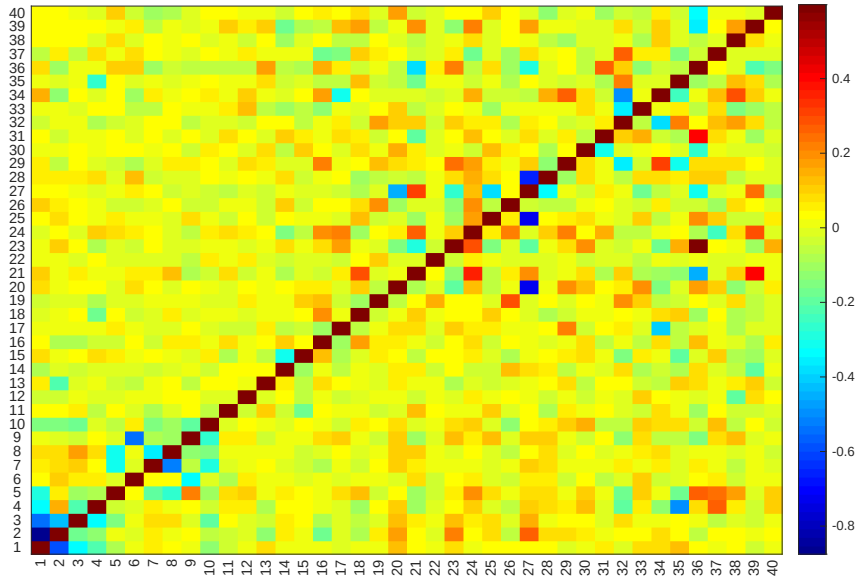


Figure 2.9 Learned weights of the last hidden layer that capture the relation between attributes (attributes order is same as in Table 2.3).

2.2.4 Conclusion

We present an end-to-end multi-task framework for face attribute classification that considers attribute location to reduce network size and correlation among attributes to improve accuracy.

We also introduce a novel ensemble technique that we call within-network ensemble, by branching output nodes from different depths of the network and computing the loss over all these branches. As the network is shared, this branching results in very little computational overhead. To the best of our knowledge, this ensemble technique has not been suggested before, while it brings non-negligible improvements (0.71% points accuracy improvement over the end-to-end network). Our results surpass state-of-the-art on both LFWA and CELEBA datasets, with 86.63% and 93.20% average accuracies, respectively.

2.2.5 Acknowledgements

We gratefully acknowledge NVIDIA Corporation with the donation of the Titan X Pascal GPU used in this research.

2.3 Relative Attribute Classification with Deep Rank SVM

Relative attributes indicate the strength of a particular attribute between image pairs. We introduce a deep Siamese network with rank SVM loss function, called Deep Rank SVM (DRSVM), in order to decide which one of a pair of images has a stronger presence of a specific attribute. The network is trained in an end-to-end fashion to jointly learn the visual features and the ranking function. We demonstrate the effectiveness of our approach against the state-of-the-art methods on four image benchmark datasets: LFW-10, PubFig, UTZap50K-lexi and UTZap50K-2 datasets. DRSVM surpasses state-of-art in terms of the average accuracy across attributes, on three of the four image benchmark datasets.

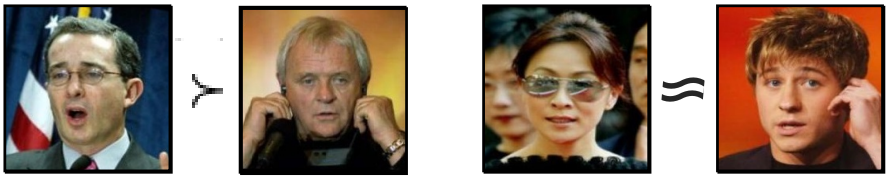
2.3.1 Introduction

Identification and retrieval of images and videos with certain visual attributes are of interest in many real-world applications, such as image search/retrieval [53, 54], video retrieval [55], image/video captioning [56, 57], face verification [7], and zero-shot learning [58, 59]. Visual attribute learning is studied in particular for binary attributes that indicate the presence or absence of a certain semantic attribute (smiley, wearing eye glasses, etc.) [18, 60].

Parikh and Grauman [61] introduced *relative attributes* with a formulation similar to that of Support Vector Machines (SVMs). The goal of relative attribute learning is to learn a function which predicts the relative strengths of a pair of images regarding a given attribute (e.g. which picture is more smiling?). The network should be able to answer the comparisons, with more/less/equal of the presence of a specific attribute. Figure 2.10 shows the comparison for two separate data sets, for the attributes *mouth-open* and *sporty* from the LFW10 and UTZap50K-2 datasets.

After the introduction of the problem, subsequent research approached the problem using either traditional features or deep learning approaches, as discussed in Section 2.3.2.

In this paper, we present a deep learning system that can compare the given two images in terms of their strength regarding a particular attribute. Specifically, we propose a convolutional Siamese network using rank SVM loss function for the relative attribute problem. The main contributions of our proposed model are summarized as follows:



(a)

(b)

Figure 2.10 Samples of visual relative attributes from the training dataset. (a) Shows random samples from LFW10 dataset of *mouth-open* attribute, and (b) Shows random samples from UTZap50K-2 dataset of *sporty* attribute.

- Proposing an end-to-end deep learning framework in which the network jointly learns the visual features and the rank SVM function, for relative attribute classification.
- Demonstrating the effectiveness of the proposed framework by comparing to our baseline [62], with improvements of 6%, 3%, 2.65%, and 1% in average ranking accuracy for LFW-10, PubFig, UTZap50K-lexi, and UTZap50K-2 datasets, respectively.
- Surpassing the state-of-the-art results in LFW-10, PubFig, and UTZap50K-lexi datasets by about 2%, 0.2%, and 0.87% and obtaining slightly lower results in UTZap50K-2 dataset.

The rest of this paper is organized as follows. A review of literature is presented in Section 2.3.2. Section 2.3.3 introduces our proposed method together using the deep rank SVM objective function. Description of the employed datasets, implementation details, along with extensive experimental results are discussed in Section 2.3.4. Finally, the paper concludes in Section 2.3.5.

2.3.2 Related Works

The relative attribute learning problem has attracted significant attention, with researchers approaching it first using traditional approaches and later using deep learning approaches. Traditional and hand-crafted features are first used in [61,

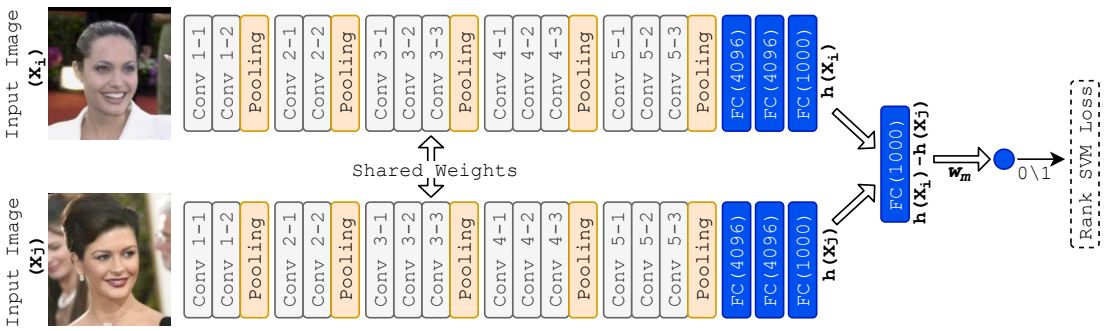


Figure 2.11 Overall Deep Rank SVM architecture. The network for a given attribute takes a pair of images ($\mathbf{x_i}$, $\mathbf{x_j}$) as input and outputs 1 if $\mathbf{x_i}$ shows the given attribute more strongly, compared to $\mathbf{x_j}$; 0 otherwise.

63, 64, 65, 66]; however, more recently, deep convolutional neural networks are used to jointly learn the features and the ranking function in an end-to-end fashion [62, 67, 68, 69].

2.3.2.1 Traditional Approaches

Parikh and Grauman [61] first proposed relative attributes where they used the GIST descriptor [70] and color histogram features, together with a constrained optimization formulation similar to that of SVMs.

Later, Li et al. [63], introduced non-linearity by using the relative forest algorithm to capture more accurate semantic relationships. More recently, Yu and Grauman [66], developed a Bayesian local learning strategy to infer when images are indistinguishable for a given attribute in a probabilistic, local learning manner.

2.3.2.2 Deep Learning Approaches

Hand-crafted feature representation may not be the best to capture the relevant visual features to describe relative attributes. As with many other visual recognition problems, deep learning approaches significantly outperform approaches that are based on hand-crafted features followed by shallow models.

Souri et al. [62] introduced RankNet, which is a convolutional neural network based on the architecture of VGG-16 [38]. A ranking layer is included to rank the strength of an attribute in the given pair of images based on the extracted features in an

end-to-end fashion.

Using a similar approach, Yang et al. [68] proposed a DRA model which consists of five convolutional neural layers and five fully connected layers followed by a relative loss function.

Singh and Lee [67] trained a Siamese network based on AlexNet [71], with a pairwise ranking loss. The network consists of two branches, each branch consists of a localization module and a ranking module. In Zhuang et al. [69], cross-image representation is considered via deep attentive cross-image representation learning (DACRL) model: an end-to-end convolutional neural network which takes a pair of images as input, and outputs a posterior probability that indicates the relative strengths of a specific attribute, based on cross-image representation learning.

Our work most resembles [62], except for the loss function. As one of our contributions is embedding the SVM loss into the network, we compare our results to this system as the baseline; as well as newer work that achieved state-of-art results [67] [69].

2.3.3 Deep Rank SVM

We introduce Deep Rank SVM (DRSVM), a convolutional Siamese network trained with the rank SVM objective function. While rank SVM formulation was proposed before [72], this is the first time it is incorporated into a deep network architecture as a loss, to the best of our knowledge.

Following Parikh and Grauman's [61] notation, training images consist of a set of ordered image-pairs $\mathbf{O_m} = (\mathbf{x_i}, \mathbf{x_j})$ and a set of un-ordered image-pairs $\mathbf{S_m} = (\mathbf{x_i}, \mathbf{x_j})$ for every attribute m of a set of M attributes. $(\mathbf{x_i}, \mathbf{x_j}) \in \mathbf{O_m}$ when the presence of attribute m in $\mathbf{x_i}$ is stronger than the presence of attribute m in $\mathbf{x_j}$ and $(\mathbf{x_i}, \mathbf{x_j}) \in \mathbf{S_m}$ when $\mathbf{x_i}$ and $\mathbf{x_j}$ have similar presence strength of attribute m.

With these notations, we can formulate the problem as learning the deep attribute representation $\mathbf{h}(\mathbf{x})$ of an image, for a specific attribute m that satisfies the following constraints:

$$\begin{aligned} \mathbf{w_m^T h}(\mathbf{x_i}) &> \mathbf{w_m^T h}(\mathbf{x_j}); \quad \forall (\mathbf{x_i}, \mathbf{x_j}) \in \mathbf{O_m} \\ \mathbf{w_m^T h}(\mathbf{x_i}) &= \mathbf{w_m^T h}(\mathbf{x_j}); \quad \forall (\mathbf{x_i}, \mathbf{x_j}) \in \mathbf{S_m} \end{aligned} \tag{2.3}$$

In this work, we use the VGG-16 architecture [38] as the base of a Siamese network

to jointly learn the deep attribute representation $\mathbf{h}(\mathbf{x})$ and the weights $\mathbf{w_m}$ to rank the two input images for the given attribute m. The network is illustrated in Figure 2.11. As seen in this figure, the output of the two branches of the network are $1,000$ dimensional each. An additional layer is added to carry out the difference between the feature representations, $\mathbf{h}(\mathbf{x_i})$ and $\mathbf{h}(\mathbf{x_j})$; followed by an output node that computes the weighted differences between the two representations. For the objective function, we use the rank SVM formulation proposed in [61]; however unlike their use of the GIST features, we aimed to jointly learn the visual features and the rank SVM function, in a deep convolutional network. Details of the whole architecture are described in Section 2.3.4.2.

The input to the rank SVM function is the deep attribute representations $\mathbf{h}(\mathbf{x_i})$ and $\mathbf{h}(\mathbf{x_j})$, computed by the Siamese network for the image-pair, $\mathbf{x_i}$ and $\mathbf{x_j}$. The rank SVM optimization function for relative attributes is defined as in [61]:

$$
\begin{aligned}
&\text{min} && \frac{1}{2}\mathbf{w_m^T w_m} + C_1 \sum \xi_{ij}^2 + C_2 \sum \gamma_{ij}^2 \\
&\text{subject to} && \mathbf{w_m^T}(\mathbf{h}(\mathbf{x_i}) - \mathbf{h}(\mathbf{x_j})) \geq 1 - \xi_{ij} \quad \forall (i,j) \in \mathbf{O_m} \\
&&& |\mathbf{w_m^T}(\mathbf{h}(\mathbf{x_i}) - \mathbf{h}(\mathbf{x_j}))| \leq \gamma_{ij} \quad \forall (i,j) \in \mathbf{S_m} \\
&&& \xi_{ij} \geq 0, \gamma_{ij} \geq 0
\end{aligned}
\tag{2.4}
$$

where $\mathbf{w_m}$ is the trainable weights of the ranking layer, the first term maximizes the margin, while the second and third terms are there to enforce the soft margin of ordered/un-ordered image-pairs on the training images. We also used quadratic terms for the soft error as in [61]. C_1 and C_2 are the trade-off constants between maximizing the margin and satisfying the pairwise relative constraints. We choose C_1 and C_2 to be equal as done in [61] and with the value of 0.1.

We then obtain the corresponding unconstrained optimization problem by combining the constraints on the slack variables ξ_{ij} and γ_{ij}, as:

$$
\begin{aligned}
\min_{\mathbf{w_m}} \quad & \frac{1}{2}\mathbf{w_m^T w_m} + C_1 \sum_{(i,j) \in \mathbf{O_m}} \max\left(0, 1 - \mathbf{w_m^T}(\mathbf{h}(\mathbf{x_i}) - \mathbf{h}(\mathbf{x_j}))\right)^2 \\
& + C_2 \sum_{(i,j) \in \mathbf{S_m}} (\mathbf{w_m^T}(\mathbf{h}(\mathbf{x_i}) - \mathbf{h}(\mathbf{x_j}))^2
\end{aligned}
\tag{2.5}
$$

As suggested in [73], the objective function is calculated with no bias term to avoid learning the constant function mapping directly to the relative ordering.

2.3.4 Experimental Evaluation

We evaluate the effectiveness of our approach on the publicly available datasets for relative attributes, described in Section 2.3.4.1. Our results are compared to the results of several systems that report accuracy as performance measurement, namely Rank SVM [61], FG-LP [64], spatial Extent [65], DeepSTN [67], DRA [68], and DACRL [69].

We consider the RankNet system proposed in [62] as our baseline. We used the same network (VGG-16 pre-trained on ILSVRC 2014) and the same data augmentation techniques, but the proposed rank SVM loss function was used instead of the cross-entropy loss used in RankNet. In this way we aimed to evaluate the effectiveness of using the SVM formulation with our deep learning framework.

In Section 2.3.4.2, the network and implementation details are explained. In Section 2.3.4.3, the performance of our proposed method is shown along with a comparison with the baseline and other state-of-the-art systems.

2.3.4.1 Datasets

Our proposed approach is evaluated on four different datasets from distinctive areas for comprehensive evaluation.

1.1 **LFW-10 Dataset [74]**: The dataset is a subset of the Labels Faces in the Wild (LFW) dataset [75]. It consists of 2,000 images with 10 different face attributes (refer to Table 2.5). For each attribute, a random subset of 500 pairs of images have been annotated for training and testing sets.

1.2 **Public Figure Face Dataset [61]**: PubFig dataset consists of 772 images from 8 different identities with 11 semantic attributes (refer to Table 2.6). The ordering of the samples are annotated at the category level; in other words, all images with the same identity are ranked higher, equal, or lower than all images belonging to another identity with respect to a specific attribute. For instance, a person is said to have longer hair than another person, even if this may not be true in all of their photographs. This short-cut in annotation will result in label inconsistencies.

1.3 **UTZap50K-2 Dataset [64]**: Large shoe dataset consists of 50,025 images collected from Zappos.com. It consists of 4 shoe attributes: open, pointy, sporty, and comfort (refer to Table 2.8). After pruning out pairs with low

confidence or agreement, the human-annotated examples consist of approximately 1,500-1,800 training image-pairs for each attribute and in total 4,334 image-pairs for testing.

1.4 **Zappos50K-lexi Dataset [76]**: It is based on UTZap50K dataset [64] with 10 additional fine-grained relative attributes: comfort, casual, simple, sporty, colorful, durable, supportive, bold, sleek, and open (refer to Table 2.7). The dataset consists of approximately 1,300-2,100 image-pairs for each attribute.

In all of our experiments, we have used the provided training/testing split by the original publishers of the datasets.

2.3.4.2 Implementation Details

We chose the VGG-16 model to be our base architecture to have a fair comparison with our baseline [62], which uses the same architecture.

The input to the model is two 224×224 RGB images similar to [62]. The output of the two branches of Siamese network $\mathbf{h}(\mathbf{x_i})$ and $\mathbf{h}(\mathbf{x_j})$ are 1,000 dimensional each, as illustrated in Figure 2.11. The weight initialization for the output node is sampled from a zero-mean Gaussian distribution with a standard deviation of 0.01.

To implement and fine-tune our architecture, we have used the pre-trained VGG-16 provided by Keras framework. The last added weights $\mathbf{w_m}$ of the ranking layer is initialized using the Xavier method without bias term. For training, stochastic gradient descent with RMSProp optimizer is used with a mini-batch size of 48 image-pairs. A unified learning rate is set to 10^{-5} for all of the layers. The training images are shuffled after every epoch.

A separate network for each attribute is trained. For LFW-10, UTZap50K-2, and UTZap50K-lexi datasets, we trained our model for 500, 200, and 200 epochs for each attribute, respectively. For PubFig dataset, the relative attributes are annotated at the category-level manner; hence one epoch contains large number of image-pairs, that is the combination of all of the images in the dataset. Therefore, we trained the model for 10,000 iterations where in every iteration a random selection of 48 image-pairs are chosen from the dataset, and ground-truth labels are assigned based on their categories.

Advanced data augmentation techniques have proven to improve performance in many studies specially for deep learning. However, to resemble our baseline and show the effectiveness of incorporating Rank SVM loss function to the deep learning, only

simple on-the-fly data augmentation techniques are applied during training, namely rotation [-15, 15], horizontal flipping, and random cropping.

2.3.4.3 Results

We compare the performance of the proposed Deep Rank SVM model (DRSVM) with our baseline [62] and state-of-the-art, on four different datasets, in Tables 2.5-2.8. The reported performance figures are accuracies over correctly ordered pairs (excluding similar pairs), in line with the literature.

Table 2.5 shows the results on the LFW-10 dataset where we outperform our baseline [62] by about 6% on average. We can attribute this to the use of the rank SVM loss as the loss function, as this is the main difference between our model and the baseline. Our results surpass the average accuracy by 1.2% points over the state-of-the-art [69], with best performance on 7 of the 10 attributes.

Table 2.6 shows the results on the PubFig dataset where our system improves over the baseline [62] by 3% points and obtains the best results on 8 out of 11 attributes compared to state-of-art [69]. The gain on this dataset is marginal (0.2%) which may be due to the category base annotation that may result in annotation inconsistencies, as discussed in Section 2.3.4.1.

Table 2.7 shows the results on the UTZap50K-lexi dataset where we outperform the baseline by 2.65%. Furthermore, we improve the average accuracy by 0.87% over the state-of-the-art [69] and obtain the best results in 6 out of the 10 attributes.

Finally, Table 2.8 shows the results on the UTZap50K-2 dataset where we outperform the baseline by 1%, but slightly underperform the state-of-art [69] by 0.15% (72.29% vs 72.44%), while obtaining best results in 2 out of 4 attributes.

Table 2.5 State-of-the-art accuracies on LFW-10 dataset compared with the results obtained in this work. Bold figures indicate the best results.

Method	Bald Head	Dark Hair	Eyes Open	Good Looking	Mascu. Looking	Mouth Open	Smile	Teeth	Fore-head	Young	Mean
FG-LP [64]	67.90	73.60	49.60	64.70	70.10	53.40	59.70	53.50	65.60	66.20	62.43%
Spatial Extent [65]	83.21	88.13	82.71	**72.76**	93.68	88.26	86.16	86.46	90.23	75.05	84.67%
RankNet [62]	81.14	88.92	74.44	70.28	98.08	85.46	82.49	82.77	81.90	76.33	82.18%
DeepSTN [67]	83.94	92.58	**90.23**	71.21	96.55	91.28	84.75	89.85	87.89	80.81	86.91%
DACRL [69] (without Attention)	83.21	91.99	87.97	69.97	97.70	89.93	85.03	88.00	89.45	74.84	85.81%
DACRL [69]	85.04	92.58	**90.23**	70.28	**98.28**	91.28	85.03	89.23	90.63	76.55	86.91%
DRSVM (proposed)	**90.75**	**92.67**	86.54	71.21	95.05	**92.67**	**88.64**	**91.80**	**90.84**	**81.02**	**88.12%**

Table 2.6 Comparison of the state-of-the-art accuracies on the PubFig dataset.

Method	Male	White	Young	Smile	Chubby	Fore-head	Bushy Eyebrow	Narrow Eyes	Pointy Nose	Big Lips	Round Face	Mean
FG-LP [64]	91.77	87.43	91.87	87.00	87.37	94.00	89.83	91.40	89.07	90.43	86.70	89.72%
RankNet [62]	95.50	94.60	94.33	95.36	92.32	97.28	94.53	93.19	94.24	93.62	94.76	94.42%
DRA [68]	90.82	87.12	91.49	92.68	89.30	94.39	90.19	90.60	91.03	90.35	91.99	90.91%
DACRL [69] (without Attention)	97.70	97.82	97.10	97.03	97.05	98.30	97.36	97.99	97.26	94.36	98.04	97.27%
DACRL [69]	96.49	97.80	**97.96**	**97.42**	**97.22**	98.05	97.48	96.91	97.74	96.83	96.27	97.29%
DRSVM (proposed)	**97.74**	**98.61**	96.32	96.14	94.47	**98.75**	**98.68**	**97.28**	**99.31**	**98.24**	**96.89**	**97.49%**

Table 2.7 Comparison of the state-of-the-art accuracies on the UTZap50K-lexi dataset.

Method	Comfort	Casual	Simple	Sporty	Colorful	Durable	Supportive	Bold	Sleek	Open	Mean
RankNet [62]	90.48	90.43	90.40	93.31	95.43	90.47	91.98	91.53	86.31	82.53	90.29%
DACRL [69] (without Attention)	**91.88**	94.44	89.93	93.01	**97.33**	92.65	92.65	91.12	89.24	87.90	92.02%
DACRL [69]	**91.88**	91.36	90.16	94.22	95.81	92.33	92.65	**92.56**	**90.71**	88.98	92.07%
DRSVM (proposed)	91.59	**95.37**	**90.91**	**96.57**	95.95	**93.31**	**94.98**	91.47	89.30	**89.99**	**92.94%**

Table 2.8 Comparison of the state-of-the-art accuracies on the UTZap50K-2 dataset.

Method	Open	Pointy	Sporty	Comfort	Mean
RankSVM [61]	60.18	59.56	62.70	64.04	61.62%
FG-LP [64]	74.91	63.74	64.54	62.51	66.43%
RankNet [62]	73.45	68.20	73.07	70.31	71.26%
DACRL [69] (without Attention)	75.45	69.80	73.78	68.54	71.89%
DACRL [69]	**75.66**	70.65	**73.87**	69.56	**72.44**%
DRSVM (proposed)	74.09	**70.90**	72.95	**71.20**	72.29%

2.3.4.4 Discussion

The reported results in Tables 2.5-2.7 show that we outperformed our baseline [62] on the four employed datasets, LFW-10, PubFig, UTZap50K-lexi, and UTZap50K-2 by 6%, 3%, 2.65%, and 1% points respectively. This shows the effectiveness of using the rank SVM loss with the deep learning approach, for the relative attribute learning problem.

Furthermore, we surpassed the state-of-art on the LFW-10, PubFig, and UTZap50K-lexi datasets by 1.2%, 0.2%, and 0.87% points and obtained slightly lower results on the UTZap50K-2 dataset (72.44% versus 72.29%).

To show the effectiveness of incorporating the rank SVM objective function into our deep learning framework, we employed the same architecture used in our baseline [62] and in DACRL [69], namely VGG-16. We expect that the performance of our model will be even higher with a more advanced network (e.g. Inception-ResNet [77] or NasNetLarge [78]) and using heavy data augmentation.

Figure 2.12 shows some images along with their output prediction values of the respective attribute. Although, our network is trained given only image-pairs, we can see that the network has learned a global ranking of a given attribute.

The trained network is able to localize on the informative regions of the image related to a given attribute, without explicitly being taught to do so during training. We calculate the derivative of the output with respect to a given input and visualize the results of the last convolutional layer as shown in Figure 2.13. The heat maps visualize the pixels in the images with the most contribution to the ranking prediction of the network.

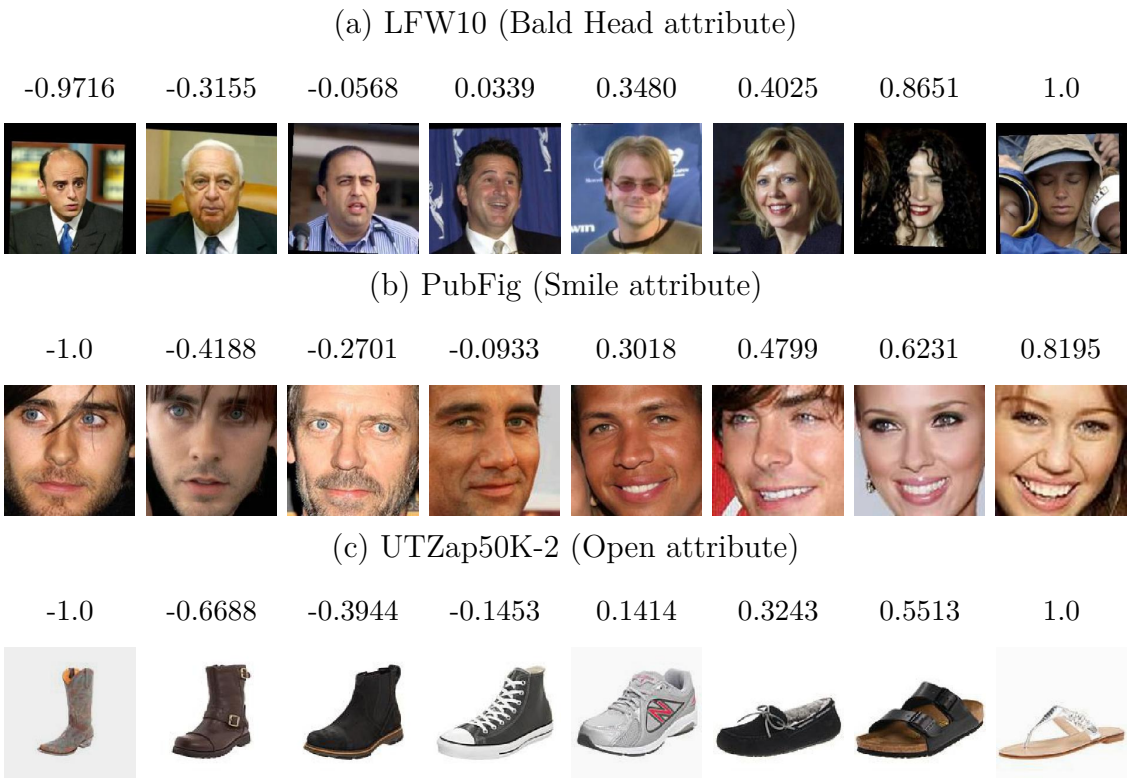


Figure 2.12 Sample images ordered according to their output prediction of their associated attribute.

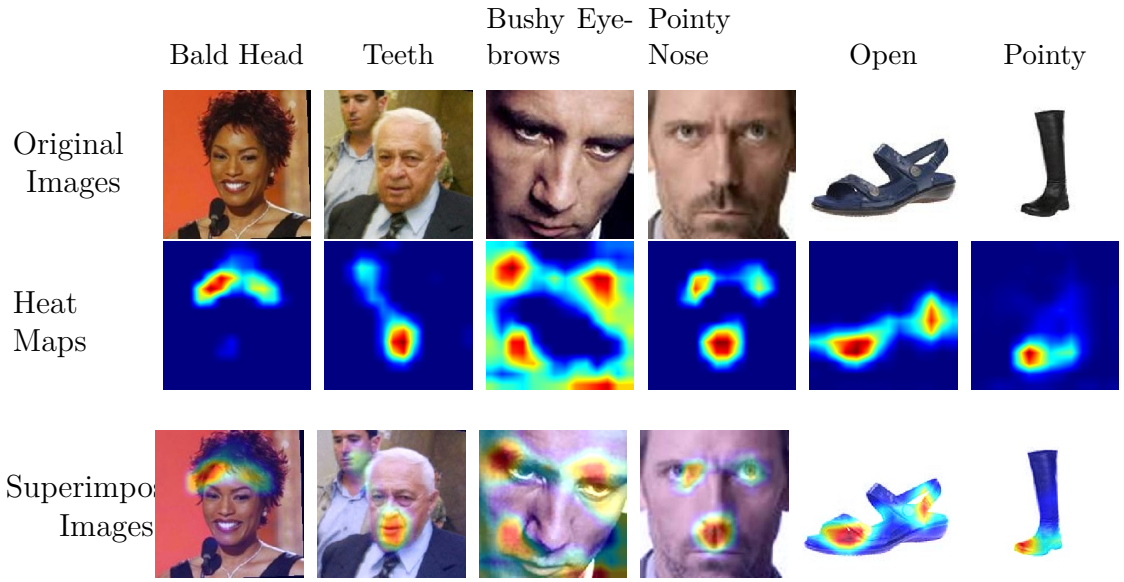


Figure 2.13 Class Activation Maps showing the pixels with most contribution to the ranking prediction in (a) Bald Head and Teeth from LFW-10 dataset, (b) Bushy Eyebrows and Pointy Nose from PubFig dataset, and (c) Open and Pointy from UTZap50K-2 dataset.

2.3.5 Conclusions and Future Work

In this paper, we proposed the deep rank SVM (DRSVM) network for relative attribute learning, to jointly learn the features and the ranking function in an end-to-end fashion. Our model is evaluated on four benchmarks, LFW-10, PubFig, UTZap50K-lexi and UTZap50K-2 and achieved state-of-the-art performance on LFW-10, PubFig, and UTZap50K-lexi datasets. These results shows the benefit of incorporating and jointly training the network with Rank SVM loss function for relative attributes.

Although the results show the ability of the network to localize on the informative regions in the image, adding a localization module similar to the one used in [69] can contribute to the performance, especially in the existence of some annotation inconsistencies, as in the case of PubFig dataset.

We believe that the performance can be further improved with a better network (e.g. Inception-ResNet [77] or NasNetLarge [78]) than the one used in this work (VGG-16), as well as using heavy data augmentation. We will add results obtained with a more powerful network in the final version of the manuscript.

Source code of the proposed method is provided in supplementary materials, and will be made public upon acceptance.

Acknowledgment

This work as supported by a grant from The Scientific and Technological Research Council of Turkey (TÜBİTAK) under project number 119E429.

3. Deep Convolutional Neural Network Ensembles Using ECOC

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.

3.1 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 [22], bagging [79], stacking [24], random forests [80], error correcting output coding [81, 23] 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 [82, 83, 25, 26, 27], or weak decision tree ensembles based on boosting deep features that have been already extracted [84, 28, 29, 30].

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 [82], 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. [83] 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 [85], 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. in [27] employ multi-resolution EfficientNets [86] 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 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) [87], light gradient boosting machine (LightGBM) [88], and categorical boosting (CatBoost) [89]. As an example of XGBoost, Pang et al. [29] 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. [90] 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. in [30] 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 [91, 92], 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 [93]. Several data-dependent and independent approaches can be used for guiding the decomposition process [94, 95]. In [96], 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 [97] 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 is demonstrated in [93, 96, 98].

Although ECOC has been commonly employed in traditional machine learning applications [99, 100, 101, 102], 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 [103], 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 [104], SVHN of Google street view images of house numbers (10 classes) [105], and PlantVillage dataset [106] consisting of 38 plant leaf disease types.

We show that for all the proposed design techniques, randECOC almost always sur-

passes 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 3.2 provides a background information on the state-of-the-art deep ensemble classification techniques as well as the ECOC framework. In Section 3.3, different ECOC training strategies using features extracted by deep convolutional neural networks are presented. This is followed by their experimental analysis in Section 3.4 and a discussion of the results obtained. Finally, conclusions of this study are presented in Section 3.5.

3.2 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 [79] is a special case of majority voting for which the base classifiers are trained on different versions [107] of the same data obtained by resampling, to ensure complementarity among the base classifiers. More complex combination rules include methods such as boosting [108], where the classifiers are trained

sequentially to compensate for the weaknesses of those already selected and stacking [109], 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 [80], 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 3.2.1, we analyse three state-of-the-art variants of the GBDT method found in the literature; namely, extreme gradient boosting (XGBoost) [87], light gradient boosting machine (LightGBM) [88], and categorical boosting (CatBoost) [89], in detail. In Section 3.2.2, 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 3.3.

3.2.1 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 [87], LightGBM [88], and CatBoost [89]. These algorithms differ from each other in terms of the mechanism used for splitting the tree nodes.

Extreme gradient boosting (XGBoost) [87], 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 [88], 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) [89] 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.

3.2.2 Error Correcting Output Coding (ECOC)

Error Correcting Output Coding (ECOC) is a generic ensemble classification framework designed for multi-class classification problems [93], 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 3.1, 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 3.1, 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.

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 [93], or problem-dependent methodologies where the aim is to split the classes in the given data domain [98, 110] meaningfully.

In decision making (testing), firstly, a given test instance $\mathbf{x}$ is classified by each base classifier to obtain the output vector $\mathbf{Y} = [y_1, ..., y_L]$ 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

$$c* = \arg \min_{i=1...k} \text{dist}(\mathbf{Y}, \mathbf{M_i}) \tag{3.1}$$

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.

Table 3.1 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

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 [96]. In practice, it has been experimentally demonstrated in [97] 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.

3.3 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 [97, 96]. 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 3.3.1 through 3.3.3, 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 3.3.1, we propose a more time-efficient implementation strategy based on multi-task learning (MTL) in Section 3.3.2. Then, in Section 3.3.3, 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 [111, 112, 113].

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.

3.3.1 Independent Learning of Base Classifiers

In this approach, the base classifiers are trained one by one and independently according to a given randECOC matrix, 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 3.1.

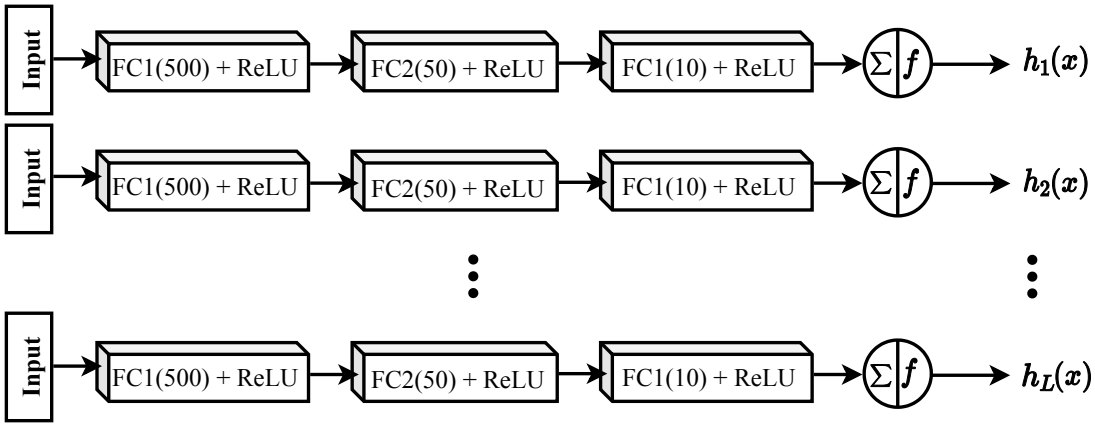


Figure 3.1 An independent base classifier architecture with a 3-hidden layer shallow network, consisting of fully connected layers followed by rectified linear units, one for each base classifier of the ECOC ensemble. The input comprises the features extracted by the bottleneck layer of a trained *base network*.

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 3.1).

3.3.2 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 3.3.1, 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 3.2.

The prediction is carried out in the same way as in Section 3.3.1, 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.

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 [17], since the nodes in the shared layers are constrained to work for multiple classifiers.

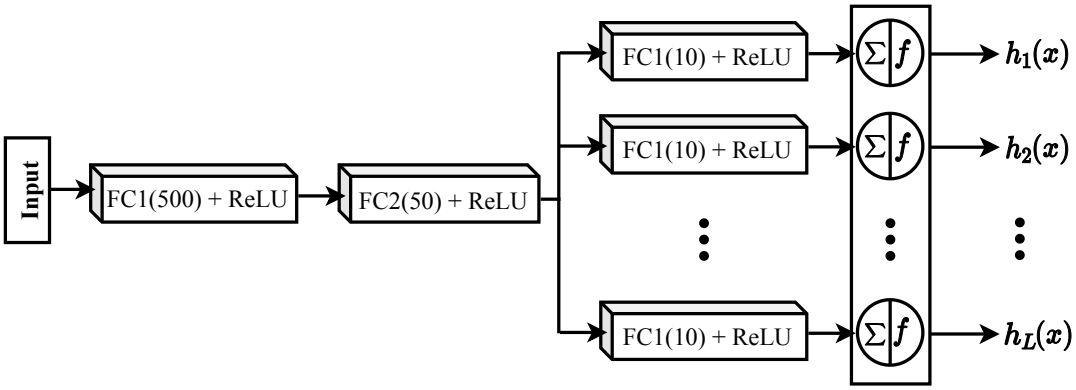


Figure 3.2 Multi-task learning architecture, with two shared modules and one classifier specific module. All layers are fully connected networks with rectified linear units.

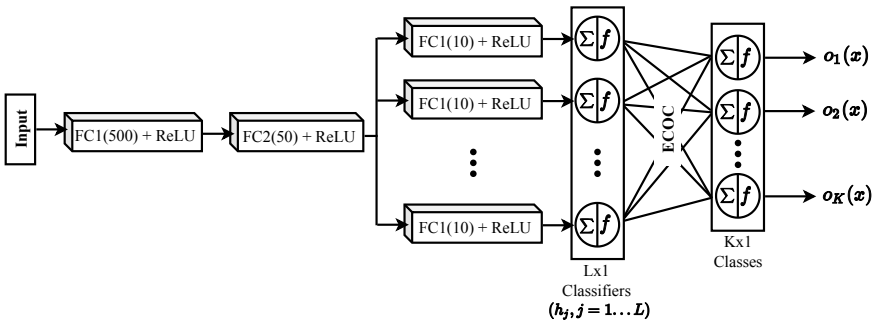


Figure 3.3 Multi-label architecture with embedded ECOC decoding, including two shared modules and one classifier specific module. The base classifier output layer is followed by the ECOC embedding layer with fixed weights. The output o_i corresponds to the score of class c_i.

3.3.3 Multi-task Learning with Embedding

Despite its advantages in terms of speed and reduced overfitting, the MTL network described in Section 3.3.2 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 3.3 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 [111, 113, 112].

Let us assume that the nodes corresponding to the base classifiers h_j, $j = 1 \dots L$ are connected to the output nodes $o_i, i = 1 \dots K$ 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

$$o_i(\mathbf{x}) = \sum_{j=1}^{L} h_j(\mathbf{x}) \times w_{ij} = \mathbf{h}(\mathbf{x}) \cdot \mathbf{w_i}. \tag{3.2}$$

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. In other words,

$$HD(\mathbf{w_c}, \mathbf{h}(\mathbf{x})) = \frac{L - o_c(\mathbf{x})}{2}. \tag{3.3}$$

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 $\mathbf{T} = [t_1 \dots t_K]$ (one-hot encoded vector where $t_c = 1$ for only the correct class and zero elsewhere), we use the loss function given in Equation 3.4. 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.

$$L = (L - o_c(\mathbf{x}))^2 + \sum_{l=1}^{L} (h_l(\mathbf{x}) - M(c,l))^2 \tag{3.4}$$

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:

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

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.

3.4 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:

2.1 Simple averaging ensemble;

2.2 Gradient boosting decision trees (GBDTs): XGBoost, LightGBM, and CatBoost;

2.3 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 3.4.1, the details of the datasets used in the experiments are presented and in Section 3.4.2, various base network architectures utilised in this study are described. This is followed by providing the details of the experimental setup in Section 3.4.3, and the thorough discussion of the results in Section 3.4.4.

3.4.1 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 [104]:** This dataset consists of 60,000 (32 x 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 [104]:** 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 [105]:** 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 [106] :** 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.

3.4.2 Base Network

To construct the base network, we employ four commonly used, state-of-the-art convolutional neural network architectures; namely MobileNetV2 [114], Inception-V3 [115], Xception [116], and Squeeze-and-Excitation Networks (SENet) [117].

MobileNetV2 proposed by Google in [114], 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 [115], 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 [116] is an extreme version and an extension of the Inception [83] 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 [117], 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.

3.4.3 Experimental Setting

All the base networks employed in this study are pre-trained on the ImageNet dataset [118]. 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 3.1, 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 3.2.

The randECOC using MTL w/ embedding, as given in Figure 3.3, 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 [119] 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 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.

3.4.4 Results

Table 3.2 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

We report the result of a comparison of the evaluated frameworks in Section 3.4.4.1 and the performance analysis of combinatory approaches in Section 3.4.4.2.

3.4.4.1 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 3.2, 3.3, and 3.4 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.

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.

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 3.2 and 3.4, respectively. It improves the testing accuracy 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. 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 [120].

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

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

Table 3.3 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

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 3.3 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.

3.4.4.2 Combinatory Approach - Ensemble Averaging of GBDT and randECOC Ensembles

As an important outcome of the comparative experiments presented in Section 3.4.4.1, 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

Table 3.4 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

GBDT and randECOC, as explained in 3.4.3.

The results of these experiments are provided in Table 3.5. 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.

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

Table 3.5 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%

Table 3.6 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%

for the CIFAR-10 dataset, while training the 5 base networks takes 3160 minutes. The additional time corresponds to 0.58% overhead.

3.4.4.3 Experiments with Real-Life PlantVillage Dataset

Experiments on PlantVillage dataset [106] 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 3.6, 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 [121, 122, 123].

3.5 Conclusions

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.

3.6 Acknowledgements

This project was partially supported by TÜBİTAK, The Scientific and Technological Research Council of Turkey, with project number 119E429.

4. Skin Lesion Diagnosis

4.1 Skin Lesion Classification With Deep CNN Ensembles

Early detection of skin cancer is vital when treatment is most likely to be successful. However, diagnosis of skin lesions is a very challenging task due to similarities between lesions from different types of cancers, in terms of appearance, location, color, and size. We present a deep learning method for skin lesion classification by fine-tuning three pre-trained deep learning architectures (Xception, Inception-ResNet-V2, and NasNetLarge) using training images provided by ISIC2019 organizers and fusing their results. Additionally, outliers and the large class imbalance are addressed to further enhance the classification performance. Experimental results show that the proposed framework obtained promising results that are comparable with the ISIC2019 challenge leader board.

4.1.1 Introduction

Computer-aided diagnostic tools have long empowered pathologists against a wide spectrum of diseases, since the first development of expert systems in the 1970s. Their performance levels have nowadays reached unprecedented levels, mostly thanks to the paradigm shifting advances in the field of machine learning, skin cancer in particular, as the most common form of this often fatal disease, has received particular attention in this regard and deep learning methods have reached a level of precision that is comparable to qualified dermatologists.

As with most diseases, early diagnosis of a particular strain of cancer is of crucial significance for the patient's successful treatment. Even though a human expert can be trained to achieve a diagnostic accuracy of skin cancer types up to approximately

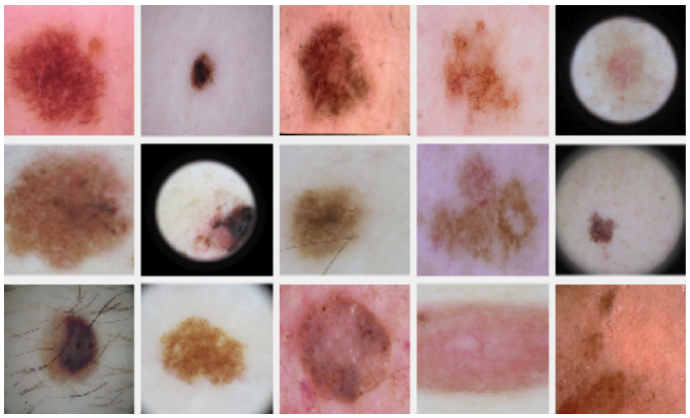


Figure 4.1 Random samples of skin lesions from ISIC2019 Training set.

80% [124], the number of dermatologists is unfortunately insufficient when compared against the disease occurrence frequency [125].

In an effort to rectify this imbalance, the International Skin Imaging Collaboration (ISIC) has developed the ISIC Archive, an international repository of validated dermoscopic images around which the ISIC challenge has been organized annually, in order to boost the development and effectiveness of appropriate computer-aided diagnostic tools.

As expected, the ISIC challenge is becoming progressively harder and more akin to real-world scenarios. This year, instead of segmentation and attribute detection tasks, the entire challenge focuses on lesion diagnosis. The dataset contains 8 strains of skin cancer (one more than 2018). Besides, the diagnostic objective has been upgraded to include a "None of the others" class as well, rendering it as an open set recognition problem. Random samples from the ISIC2019 dataset are shown in Figure 4.1.

This paper presents the developed system for the ISIC2019 challenge, and details our findings. Our system relies on an ensemble of various modern convolutional neural networks varying from each other in terms of architecture, preprocessing and data augmentation techniques. Furthermore, a comprehensive study of fusion strategies has been conducted, further supported by state of the art gradient boosting methods. Finally, special precautions have been taken for anomaly detection, so as to handle the case of samples stemming from unknown classes. Our proposed method obtained promising results that are comparable with the ISIC2019 challenge leader board [1].

The rest of this paper is organized as follows: Section 4.1.2 describes the developed system based on the fine-tuning of Xception, Inception-ResNet-V2, and NasNet-

[1] https://challenge2019.isic-archive.com/leaderboard.html

Large models for skin lesion classification. Next, Section 4.1.3 is dedicated to the description of the utilized dataset, data augmentation, and classifiers' fusion and presentation of designed experiments and their results. The paper concludes in Section 4.1.4 with a summary and discussion of the utilized methods and obtained results.

4.1.2 Skin Lesion Classification

In recent years, there have been many breakthroughs in the development of deep learning using Convolutional Neural Networks (CNN). In this work, we tackled the skin lesion classification problem using three of the latest and most accurate models, namely Xception [116], Inception-ResNet-V2 [77], and NasNetLarge [78].

Xception [116] is an extreme version and an extension of the Inception [83] architecture, which replaces the standard Inception modules with depth-wise separable convolutions. The network is 71 layers deep with only 22.9 million parameters and an image input size of 299-by-299. Inception-ResNet-V2 [77] is an advanced convolutional neural network that combines the inception module with ResNet [51] to increase the efficiency of the network. As for NasNetLarge [78], authors propose to search for an architectural building block on a small dataset and then transfer the block to a larger dataset. Initially, they search for the best convolutional layer on CIFAR-10, then apply this layer to ImageNet by stacking together more copies of this layer. They also proposed a new regularization technique called Scheduled-DropPath that significantly improves the generalization of their proposed network.

Our approach is based on fine-tuning and fusing of the aforementioned three successful deep learning models. These three models are currently the top-ranked architectures of the ImageNet Large-Scale Visual Recognition Challenge (ILSVRC) 2014. The models are pre-trained on the ILSVRC 2012 dataset with 1.2 million labeled images of 1,000 object classes. Different network configurations are used to further handle the class imbalance. The distribution of the eight given categories of the ISIC dataset is shown in Figure 4.2.

Ensemble learning techniques have seen a huge jump in popularity in the last years. Ensemble can help in building a much robust model from a few weak models, which eliminates a lot of the model tuning that would otherwise be needed to achieve good results. In this work, we used LightGBM [88], one of the most famous ensemble techniques nowadays.

LightGBM is an open-source framework which trains a Gradient Boosted Decision

Tree (GBDT). In GBDT, successive models are found by applying gradient descent in the direction of the average gradient, calculated with respect to the error residuals of the loss function of the leaf nodes of previous models. In this work, we trained LightGBM using the extracted features from the last pooling layer of our trained models.

All training and testing were conducted on a Linux system with a Titan X Pascal GPU and 12GB of video memory.

4.1.2.1 Anomaly Detection

The goal of the ISIC2019 competition is to classify dermoscopic images among nine different diagnostic categories while only eight classes are given for training. One way of dealing with the unknown class is to consider all of the instances coming from this class as outliers and target them using one-class learning approaches. One-class learning is a challenging task especially when dealing with high dimensional data points. In this paper, we applied one-class learning using deep neural network features and compared classifier performance based on the approaches of OC-SVM [126], Isolation Forest [127], and Gaussian Mixtures [128] as shown in Section 4.1.3. We found that the best approach for this dataset is Isolation Forest [127]. Isolation Forest is based on the fact that the features of anomalies are very different from the normal samples. The idea is to build an ensemble of isolation trees where anomalies have short average path lengths on the those trees.

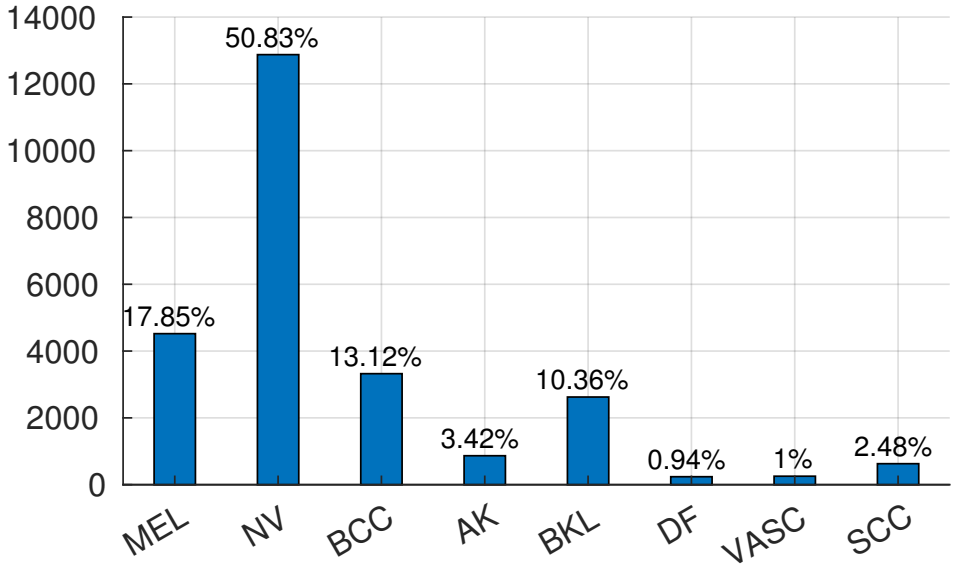


Figure 4.2 Distribution of the available ISIC2019 training images across the eight given skin lesion categories.

4.1.3 Experiments and Results

The training data of ISIC2019 includes skin lesion images from several datasets, such as: HAM10000 [129], BCN20000 [130], and MSK [131] datasets. The goal of ISIC2019 is to classify dermoscopic images among nine different diagnostic categories: 1. Melanoma (MEL); 2. Melanocytic nevus (NV); 3. Basal cell carcinoma (BCC); 4. Actinic keratosis (AK); 5. Benign keratosis (BKL); 6. Dermatofibroma (DF); 7. Vascular lesion (VASC); 8. Squamous cell carcinoma (SCC); and 9. None of the others (UNK). The dataset consists of 25,331 images for training across 8 different categories. Furthermore, the test dataset contains an additional outlier class that is not represented in the training data.

Two tasks are available for this competition: 1) classify dermoscopic images without meta-data, and 2) classify images with additional available meta-data. In this paper, we target the first task where only the provided images are used without any usage of meta-data or external dataset.

4.1.3.1 Ensemble of Deep Neural Networks

Generally, neural networks have high variance due to the stochastic training approach that make them sensitive to the nature of the training data. The models may find a different set of weights each time they are trained, which in turn may produce different predictions.

A successful approach to reduce the variance of neural network models is ensemble learning, where multiple models are trained instead of a single model and then combining the predictions from these models. Not only this approach reduces the variance of the predictions but also can result in predictions that are better than any single model.

Therefore, we trained several convolutional neural network models to tackle this problem. At first, we split the training set into 80-20% ratio to create the validation set to fine-tune the learning rate. We found that the best learning rate for the three used models, Xception, Inception-ResNet-V2, and NasNetLarge is 0.01 with validation accuracy around 90%.

We implemented Xception, Inception-ResNet-V2, and NasNetLarge models using Matlab's Deep Learning Toolbox. All the weights were fine-tuned from the pre-trained weights on the ImageNet dataset, while the last layer was learned from

Table 4.1 Specifications of the trained CNN models

Architecture	**Specifications**		
	Batch Size	**# of Epochs**	**Loss Function**
Xception	32	200	Cross Entropy
	32	30	Focal Loss with $\gamma = 1$
	32	30	Focal Loss with $\gamma = 2$
	32	80	Focal Loss with $\gamma = 3$
	32	30	Focal Loss with $\gamma = 4$
Inception-ResNet-V2	20	50	Cross Entropy
	32	70	Cross Entropy
	64	90	Cross Entropy
NasNetLarge	20	25	Cross Entropy

scratch. We used the same learning rate (0.01) for all of the systems.

During training, several data augmentation techniques were applied, such as heavy rotation [−90 to 90], x and y translation [−10 to 10], vertical and horizontal flipping. All data augmentation were applied on the fly, which means, at every iteration, different setting of augmentations are applied on top of the original batch of images.

The specifications of the trained models are shown in Table 4.1 where our systems are trained with different batch sizes and different number of epochs. Also, we employed different loss functions, namely, cross entropy and focal loss. Focal loss function (Equation 4.1) is used to address the imbalance between classes. Various Xception networks were trained with α_i set to the inverse class frequency and several values of γ_i as shown in Table 4.1.

$$FL(p, y) = -\sum_i \alpha_i y_i (1 - p_i)^\gamma \log(p_i) \tag{4.1}$$

where p_i and y_i are the prediction and the ground-truth of a given sample, respec-

tively.

Finally, in testing time, we applied test time augmentation (TTA). Specifically, we applied rotation with 90, 180, 270 degrees with and without horizontal flipping to have 6 augmented images. In addition, we applied 30 random augmentations similar to the techniques applied during training but with a smaller rotation range: [-15, 15]. To further boost the efficiency and reduce the variance, we trained a LightGBM module using the extracted features of the last pooling layer of each trained model. Score-level averaging is applied to combine the prediction scores assigned to each class for all the augmented patches; locally, within a single network and globally, among different models. The probability of the UNK class is set to $1 - max$ of the probabilities of the other eight classes for a given sample.

Figure 4.3 shows the ROC curve that is plotted with true positive rate against the false positive rate of each lesion category, individually. Table 4.2 shows the performance comparison of our model to the top two ranking results in the ISIC2019 challenge leader board of the eight given classes of skin lesion classification task. It shows that our approach surpassed top-2 rank with a high margin equals to 0.1246 and achieved comparable results with the top-1 rank method, despite usage of external data.

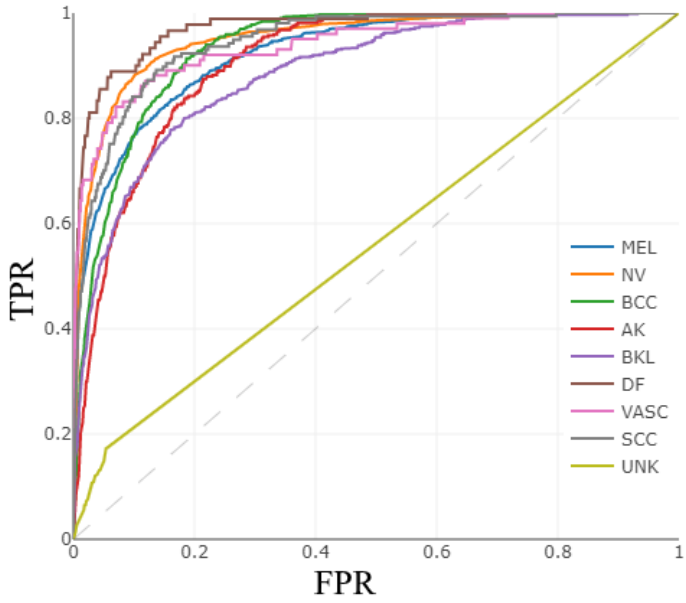


Figure 4.3 ROC curve of the nine skin lesion categories using deep CNNs ensembles.

Table 4.2 Performance of our model to the top two ranking results on ISIC2019 leader board.

Method	External Data?	MEL	NV	BCC	AK	BKL	DF	VASC	SCC	**Mean**
		AUC (Area Under the Curve)								
Top-1 Rank	Yes	0.928	0.960	0.949	0.914	0.904	0.979	0.956	0.938	0.9410
Top-2 Rank	No	0.808	0.878	0.868	0.765	0.762	0.832	0.797	0.744	0.8067
Ours	No	0.925	0.951	0.934	0.902	0.885	0.968	0.941	0.944	0.9313
		Accuracy								
Top-1 Rank	Yes	0.900	0.889	0.912	0.940	0.934	0.987	0.986	0.975	0.9404
Top-2 Rank	No	0.896	0.902	0.888	0.916	0.927	0.982	0.984	0.962	0.9321
Ours	No	0.903	0.894	0.873	0.945	0.923	0.989	0.989	0.980	0.9370

As for the unknown class in the ISIC dataset, we addressed it by the notion of anomaly detection. We applied one class learning using the features of the last pooling layer from the trained networks as they are considered to be more powerful representations of the images than handcrafted features.

We tried several one-class learning approaches like one-class support vector machines (OC-SVM), Isolation Forest and Gaussian Mixtures. We found that Isolation Forest is the best approach to be used in this task.

To show the empirical effectiveness of this step, iteratively, one class from the eight given classes is chosen to be an outlier and removed from the training procedure. In other words, for each experiment, we set the validation set to be all of the samples belonging to the anomaly class in addition to 20% from the other classes and the rest are left for training. The performance of Isolation Forest is shown in Table 4.3.

Table 4.3 Performance of Isolation Forest using deep learning features extracted from the last pooling layer of one of the trained Xception network.

Anomaly Class	**Validation Accuracy**	**Precision**	**Recall**
Class 1	92.76%	94.72%	89.88%
Class 2	97.16%	88.13%	90.07%
Class 3	94.08%	98.14%	89.81%
Class 4	90.01%	97.38%	90.08%
Class 5	91.26%	94.25%	89.97%
Class 6	90.64%	99.77%	90.26%
Class 7	90.78%	100%	90.16%
Class 8	90.05%	98.67%	89.58%

To incorporate isolation forest into our approach, we used the features of the whole training set coming from the last pooling layer of our trained models as an input to isolation forest. In testing time, we assigned the probability of the UNK class to the probability coming from isolation forest. As shown in Figure 4.4, The ROC curve of the UNK class indicates the effectiveness of adding anomaly detection to our approach compared to the ROC curve of UNK class in Figure 4.3 to improve the prediction of the UNK class.

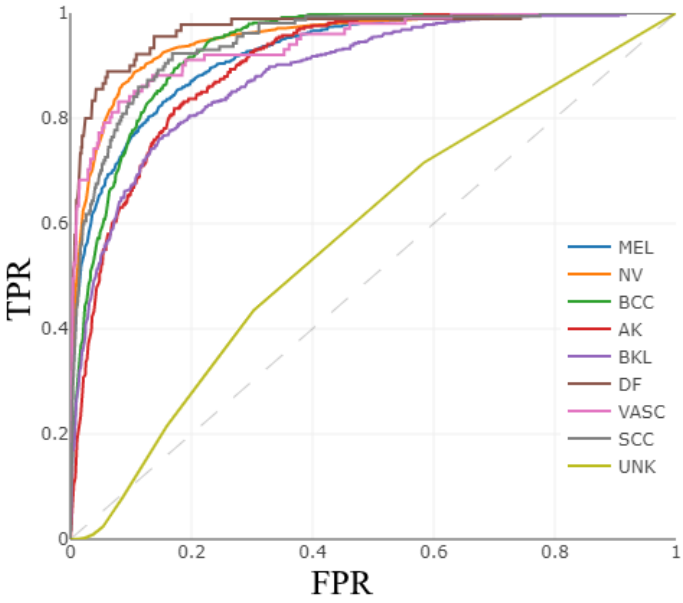


Figure 4.4 ROC curve of the nine skin lesion categories after incorporating anomaly detection.

4.1.4 Conclusions

The core of our approach is based on an ensemble and fusing of three pre-trained deep learning architectures (Xception, Inception-ResNet-V2, and NasNetLarge) using training images provided by the ISIC2019 organizers. LightGBM and one class classification models are used to further boost our predictions. In future work, we would like to investigate deep learning approaches for anomaly detection.

4.2 Skin Lesion Diagnosis With Imbalanced ECOC Ensembles

Diagnosis of skin lesions is a challenging task due to the similarities between different lesion types, in terms of appearance, location, and size. We present a deep learning method for skin lesion classification by fine-tuning three pre-trained deep learning architectures (Xception, Inception-ResNet-V2, and NasNetLarge), using the training set provided by ISIC2019 organizers. We combine deep convolutional networks with the Error Correcting Output Codes (ECOC) framework to address the open set classification problem and to deal with the heavily imbalanced dataset of ISIC2019. Experimental results show that the proposed framework achieves promising performance that is comparable with the top results obtained in the ISIC2019 challenge leaderboard.

4.2.1 Introduction

Early detection of cancer is vital for successful treatment. Even though human experts can achieve a diagnostic accuracy of approximately 80% for classifying different skin cancer types [124], the number of dermatologists is insufficient when compared against the frequency of the disease occurrence [125]. On the other hand, the performance of computer-aided diagnostic tools has reached unprecedented levels in recent years. In particular, deep learning systems have reached a level of precision that is comparable to qualified dermatologists in classifying skin cancers [132, 133].

As an effort to support clinical training and boost the performance of automated systems, the International Skin Imaging Collaboration (ISIC) has developed the ISIC Archive, an international repository of validated dermoscopic images, to enable the automated diagnosis of melanoma from dermoscopic images. The ISIC challenge has been organized annually since 2016, to measure performance of automated systems on this task [134].

In previous years, the ISIC challenge aimed several sub-tasks related to the detection, segmentation and classification of skin lesions. This year, the ISIC2019 challenge focuses on lesion classification. The dataset contains 8 classes of skin cancer, while the evaluation is done as an open-set problem where images belonging to other cancer types need to be classified as the 9th class ("None of the others").

This paper presents the system we developed for the ISIC2019 challenge and details our findings. Our system is based on a set of various state-of-art Convolutional

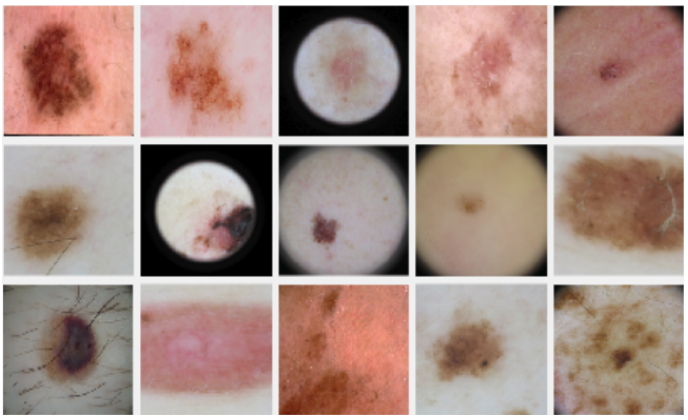


Figure 4.5 Random samples of skin lesions from ISIC2019 Training set.

Neural Networks (CNNs), varying from each other in terms of architecture, preprocessing, training configurations, and data augmentation techniques. We combine deep convolutional networks with the Error Correcting Output Codes (ECOC) framework [135], to address the open set classification problem and to deal with the heavily imbalanced dataset. To the best of our knowledge, the proposed framework (ImbECOC) is novel and the end-to-end architecture has many advantages. Several task-specific data augmentation techniques are performed for further enhancement.

The rest of this paper is organized as follows. Section 4.2.2 describes the developed system based on the fine-tuning of Xception, Inception-ResNet-V2, and NasNet-Large models for skin lesion classification and the incorporation of ECOC framework. Description of the utilized dataset, data augmentation, and classifier fusion, along with the experiments and results are discussed in Section 4.2.3. The paper concludes in Section 4.2.4 with a summary and discussion of the utilized methods and obtained results.

4.2.2 Skin Lesion Classification with ECOC ensemble

We adopt a deep learning approach, as used in many computer vision problems in recent years [136, 137, 138], with a special focus on two challenging aspects of the skin lesion classification problem: the open-set classification with the unknown class and the imbalance among the classes. Starting from multiple convolutional networks that are fine-tuned for this problem, we adopt the Error Correcting Output Codes (ECOC) approach, with the aim of addressing these two problems.

In our previous studies, we had tried training ECOC ensembles with deep features obtained from a convolutional network, as a quick and efficient way of obtaining deep learning ensembles, without much success in terms of accuracy [139]. In this

work, the initial motivation was to use the decoding distance as a way to identify unknown samples. Later on, we added a new term in the loss function to address the class imbalance. Note that the end-to-end nature of the proposed ImbECOC model allows for the use of error correction together with a new loss function.

Our approach is composed of 3 elements: (i) fine-tuning deep learning models as the base of our ensemble approach (ii) building an ECOC model from each of these base networks (iii) fusing them by simple ensemble averaging. In Section 4.2.2.1, we describe the use of transfer learning to train our base convolutional neural networks, and in Section 4.2.2.2, we describe the use of ECOC codes to construct an ensemble from each trained base network.

4.2.2.1 Base models

We use three of the top-ranked architectures trained on the ImageNet Large-Scale Visual Recognition Challenge (ILSVRC) in 2014 [140]: Xception [116], Inception-ResNet-V2 [77], and NasNetLarge [78].

The first network, Xception [116], is an extension and an extreme version of the Inception [83] architecture, which replaces the standard Inception modules with depth-wise separable convolutions. The network is 71 layers deep with only 22.9 million parameters. The image input size of the network is 299-by-299.

The second network, Inception-ResNet-V2 [77], is an advanced convolutional neural network that combines the inception module with ResNet [51] to increase the efficiency of the network. The network is 164 layers deep with only 55.9 million parameters. The image input size of the network is 299-by-299.

The third network we used, NasNetLarge [78], is built by training an architectural building block on a small dataset (e.g. CIFAR-10) and transfering it to a larger dataset by stacking together more copies of this block. Along with this new architecture, a new regularization technique called ScheduledDropPath is proposed that significantly improves the generalization of the proposed network. The image input size of the network is 331-by-331.

The models are pre-trained on the ILSVRC 2012 dataset with 1.2 million labeled images of $1,000$ object classes and fine-tuned on the ISIC dataset. The distribution of the 8 given classes of the ISIC dataset is shown in Table 4.5. All training and testing were conducted on a Linux system with a Titan X Pascal GPU and 12GB of video memory. The network configurations are given in Table 4.4 and details of

the different configurations are discussed in Section 4.2.3.2.

Table 4.4 Specifications of the trained CNN models.

#	Model	Specifications		
		Batch Size	**# Epochs**	**Loss Function**
1	Xception	32	100	Cross Entropy
2	Xception	32	100	Focal Loss ($\gamma = 3$)
3	Inception-ResNet-V2	64	60	Cross Entropy
4	Inception-ResNet-V2	64	60	Focal Loss($\gamma = 3$)
5	Inception-ResNet-V2 (Shades of Gray)	64	60	Cross Entropy
6	NasNetLarge	20	25	Cross Entropy

Neural networks can have high variance due to the stochastic training approach: the models may find a different set of weights each time they are trained, which may produce different predictions. Furthermore, deep learning systems can overfit due to the large number of parameters, especially when dealing with small training data. Therefore, for the ISIC2019 challenge, we use an ensemble of 6 ImbECOC models that are obtained as described in Section 4.2.2.2.

4.2.2.2 The ImbECOC Framework

Ensemble learning techniques have been studied theoretically and experimentally, in the last 20 years [141], both in the context of obtaining strong classifiers from weak ones and also to improve performance even with highly accurate deep learning systems. They have been shown successfully in many machine learning and computer vision tasks [142, 143, 144].

In this work, we used an Error-Correcting Output Codes (ECOC) approach as the ensemble method to be used with convolutional neural networks.

In addition, we propose a novel approach to address the large imbalance in the ISIC dataset by changing the loss function of the ECOC ensemble,

Standard ECOC Framework: The main idea behind ECOC is to decompose the multi-class classification problem into several binary sub-problems. Standard ECOC has 3 stages: encoding, learning, and decoding.

In the encoding phase, given a multi-class classification problem with k classes, an encoding matrix $\mathbf{M}$, also called the *code matrix*, is often randomly generated to specify the binary subproblems for the ensemble. Each column M^j of the matrix indicates the desired output for the corresponding binary classifier of the ensemble. Each row M_i corresponds a unique codeword of length l for the corresponding class i, indicating the desired output for that class by each of the binary classifiers, also called the *base classifiers* in the ensemble.

In the learning phase, l independent binary classifiers are trained according to the given code matrix. Specifically, a base classifier $h_j, j = 1, \ldots l$ is learned according to the column M^j of the code matrix.

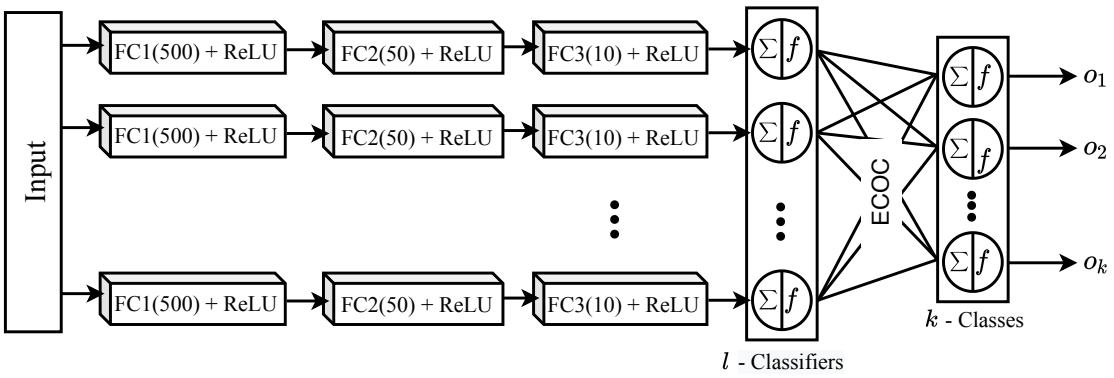


Figure 4.6 ImbECOC architecture

In decoding phase, a given test instance $\mathbf{x}$ is classified by each base classifier to obtain the output vector $\mathbf{Y} = [y_1, ..., y_l]$ where y_i is the hard or soft output of the classifier h_i for input $\mathbf{x}$. Then, the distance between $\mathbf{Y}$ and the *codeword* $\mathbf{M_i}$ of class c_i is computed, typically using the Hamming distance. The class $c*$ associated with the minimum distance is chosen as the predicted class, such that

$$c* = \arg \min_{i=1...k} \text{dist}(\mathbf{Y}, \mathbf{M_i}) \tag{4.2}$$

ImbECOC Model: We use the ECOC framework in combination with the base networks described in 2.1, to obtain the ImbECOC model, which is designed to address the open-set classification problem and the class imbalance issue.

The input to ImbECOC consists of deep features extracted from the last pooling layer of a base network. Using this input, the base classifiers of the ECOC model are trained simultaneously, using a multi-task learning approach, within a single network. Finally, an output layer whose weights are obtained from the ECOC matrix, is added, so as to directly map the input to the classes. In this way, we

obtain an end-to-end network that can be trained with a variety of loss functions. The network architecture is shown in Figure 4.6.

The output o_i of ImbECOC, corresponding to class i is the dot product between the output $\mathbf{Y}$ of the base classifiers of the ECOC and the codeword $\mathbf{M_i}$ of class i. This output ranges in $[-l, l]$ and is maximum when the output of all l base classifiers match the desired codeword of the input. The output is minimum when the base classifiers‘ output are totally the opposite of the codeword.

The loss function for the correct class c is shown in Equation 4.3, where x is the input; y_i is the output of the base classifier h_i; M(c,i) is the ECOC matrix element for the correct class and the ith base classifier and o_c is the output of the correct class:

$$Loss = \sum_{i=1}^{l} (y_i - M(c,i))^2 + (o_c - l)^2 \tag{4.3}$$

The first term penalizes difference between the prediction the base classifiers and the corresponding codeword entries for the correct class. The second term is used to maximize the output o_c corresponding to the correct class.

The standard ECOC aims to maximize the accuracy of the binary classification task regardless of the importance of the input sample. This makes standard ECOC inadequate to deal with imbalanced multi-class data. To address the imbalanced class issue, we added the normalized inverse frequency of each class, W_c, to the loss function as follows:

$$Loss = \sum_{i=1}^{l} (y_i - M(c,i))^2 + W_c \times (o_c - l)^2 \tag{4.4}$$

where W_c is the inverse frequency of class c. By incorporating W_c to the loss function, the optimal weights are obtained by minimizing a weighted loss in favor of the minority classes.

During testing, the posterior probability of each class i given the input x ($P(c_i|x)$) is directly obtained by normalizing the output of the ImbECOC network The normalization is done by omitting the negative values (least probable classes) and dividing by l, which is the maximum value that can be obtained, to normalize the probabilities to range between 0 and 1. Low confidence scores are used in detecting unknown samples, which was the original motivation for the use of ECOC in this work.

4.2.3 Experiments and Results

4.2.3.1 Dataset and Problem Definition

The training data of ISIC2019 includes skin lesion images from several datasets, such as: HAM10000 [129], BCN20000 [130], and MSK [131] datasets. The dataset consists of $25,331$ images for training across 8 different categories, while the test set contains an additional outlier (unknown) class. The goal of ISIC2019 competition is to classify dermoscopic images among these nine categories, as shown in Table 4.5.

Table 4.5 Distribution of the available ISIC2019 training images across the 8 given skin lesion categories.

Diagnostic	**# Images**	**Ratio**
Melanoma (MEL)	4522	17.85%
Melanocytic nevus (NV)	12875	50.83%
Basal cell carcinoma (BCC)	3323	13.12%
Actinic keratosis (AK)	867	3.42%
Benign keratosis (BKL)	2624	10.36%
Dermatofibroma (DF)	239	0.94%
Vascular lesion (VASC)	253	1%
Squamous cell carcinoma (SCC)	628	2.48%
None of the others (UNK)	0	0%

Two tasks are available for this competition: 1) classify dermoscopic images without meta-data, and 2) classify images with additional available meta-data. In this paper, we target the first task where only the provided images are used, without using the meta-data or any external dataset.

4.2.3.2 Base Networks

In order to construct an ensemble, we train several convolutional neural network models that are differentiated by varying the input channels, batch sizes, loss functions, and training durations.

For the input of the models, we feed the employed $CNNs$ with the RGB images except for one model where Shades of Gray [145] color constancy method is applied

to the images before training with Minkowski norm $p = 6$ as suggested in [146]. For the loss functions, we evaluated the cross entropy and focal loss. In particular, the focal loss function given in Equation 4.5, is used to address the imbalance between classes.

$$FL(p, y) = -\sum_{i} \alpha_i y_i (1 - p_i)^{\gamma} \log(p_i) \tag{4.5}$$

where p_i and y_i are the prediction and the ground-truth of a given sample, respectively. The value of α is set to the inverse class frequency. Lastly, the value of γ is finetuned. The highest validation accuracy is obtained when γ is equal to 3 in all of our experiments. The training configurations that are used are summarized in Table 4.4.

RMSPROP optimizer is used for training all of the CNN models with an initial learning rate of 3e−3. The learning rates were selected based on the validation accuracy.

4.2.3.3 ImbECOC Model

The architecture of the ImbECOC model is shown in Figure 4.6. It is trained on top of each CNN trained model, using the extracted features of the last pooling layer of that model as input.

Every branch in the model consists of 3 hidden layers with 500, 50, and 10 units. The first 2 hidden layers are followed by a dropout layer and a Rectified Linear Unit (ReLU) activation function. Third layer is followed by only ReLU activation layer. Output layer consists of l units and followed by a hyperbolic tangent (Tanh) layer.

We used the RMSPROP optimizer for training ImbECOC with an initial learning rate of 3e−4 and batch size of 512 samples.

4.2.3.4 Data Augmentation

To build a powerful deep learning classifier, large and quality training dataset are essential. Data augmentation plays a crucial role in avoiding overfitting and increasing the number of training images specially when number of images of different

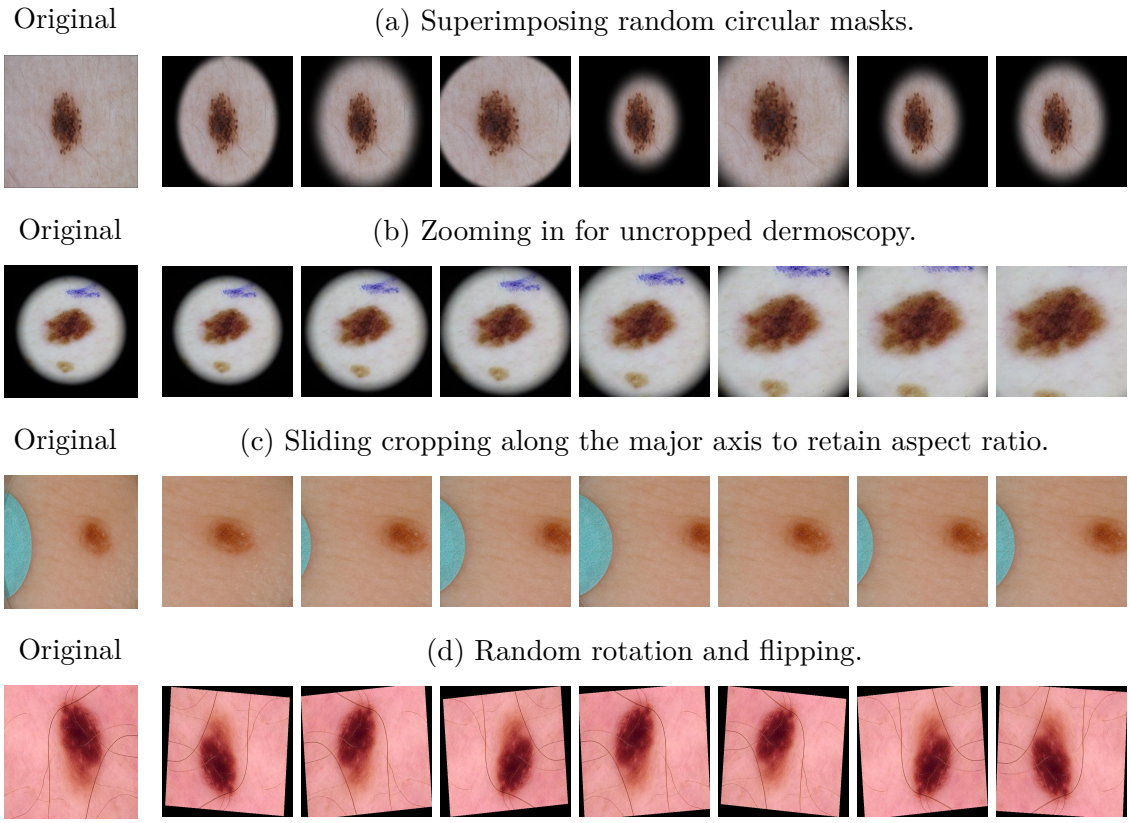


Figure 4.7 Random augmented samples from ISIC2019 training set.

categories varies widely.

In this work, we applied several commonly used data augmentation techniques during training, such as rotation [−10 to 10], x and y translation [−5, 5], and vertical and horizontal flipping. Random augmented samples are shown in Figure 4.7. Three task-specific data augmentation techniques are also proposed for further enhancement, as described below.

The ISIC2019 dataset consists of a large number of *uncropped* dermoscopy images, where black area surrounds the lesion. As one task specific data augmentation technique, we randomly superimposed circular masks to the images during training (top row of Figure 4.7). Particularly, we multiplied a circular mask positioned in the center of the dermoscopy image with random radius followed by a Gaussian filter.

A second augmentation technique is used to eliminate the dark circle around the uncropped dermoscopy[2]. For this, several crops with random sizes are performed around the center of the image (second row from the top in Figure 4.7).

Lastly, to retain the aspect ratio and avoid deformation of the dermoscopy, we feed the network with random square cropping along the major axis of the dermoscopy

[2]We consider a dermoscopy as uncropped if one-fourth the number of original pixels are dark.

images (third row from the top in Figure 4.7).

Random samples of the superimposed images, zooming in, and slide cropping strategies are shown in Figure 4.7, where the first image in each row represents the original image.

All data augmentation were applied on the fly, which means that, at every iteration, different setting of augmentations are applied on top of the original batch of images. Finally, in testing time, we applied test time augmentation. Specifically, we applied 50 random augmentations similar to the techniques applied during training.

4.2.3.5 Results and Discussion

For the evaluation, augmented test images are passed to the 6 trained deep learning models and test features extracted from the last pooling layer from each model. These are then fed to the trained ImbECOC models. Score-level averaging is applied to combine the prediction scores assigned to each class for all the augmented patches, within a single ImbECOC model; and also globally, among the different ImbECOC models.

Table 4.6 Validation and testing accuracies across different models. The reported accuracy is the normalized multi-class accuracy.

Model	Validation Accuracy		Testing Accuracy	
	CNN	w/ ImbECOC	CNN	w/ ImbECOC
Model 1	0.674	0.808	0.480	0.497
Model 2	0.657	0.670	0.486	0.514
Model 3	0.809	0.798	0.498	0.499
Model 4	0.634	0.797	0.512	0.519
Model 5	0.715	0.744	0.475	0.495
Model 6	0.789	0.851	0.512	0.521
Ensemble	0.837	0.904	0.553	0.602

The maximum probability obtained by any of the k classes, *maxscore*, is calculated and used in rejecting a sample (to label it as UNK). The probability of the UNK class is set to $1 - maxscore$ if *maxscore* is less than 0.3 (indicating that the agreement between the output vector and the closest codeword is less than 30%. If *maxscore* is higher than 0.3, the sample is accepted to belong to one of the 8 classes and the probability of the UNK class is set to 0. The 0.3 threshold is found experimentally.

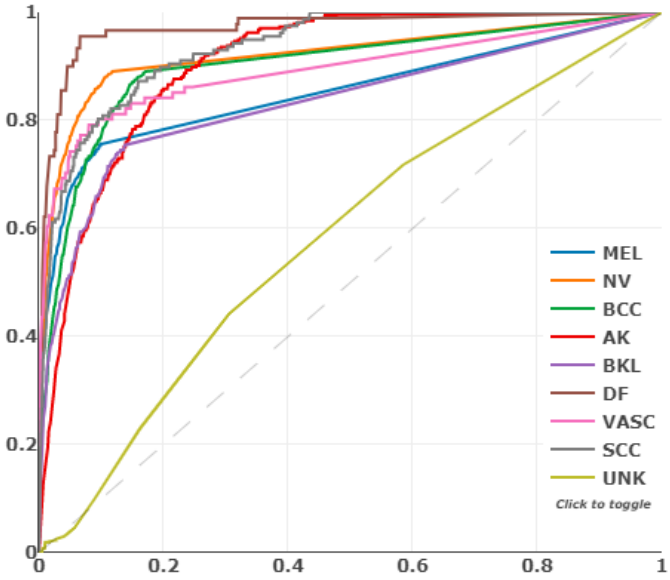


Figure 4.8 ROC curve of the 9 skin lesion classes using deep CNNs ensembles and ImbECOC.

Following the aforementioned strategy, the validation and testing accuracies with and without ImbECOC are shown in Table 4.6. Considering these results, we see that simple ensemble averaging improves over the individual networks, obtaining 55.3% normalized multi-class accuracy and ImbECOC further improves the normalized multi-class accuracy by 4.9% points, reaching 60.2%.

The ROC curve (true positive rate against the false positive rate) of each lesion category, individually is shown in Figure 4.8.

The system described in this paper has the 2nd highest ranked results in the live competition at the time of paper submission, *among teams that do not use external data*[3].

4.2.4 Conclusions

The core of our approach is based on an ensemble of deep networks, based on three separate pre-trained deep learning architectures (Xception, Inception-ResNet-V2, and NasNetLarge), trained using training images provided by the ISIC2019 organizers.

ImbECOC approach is presented to address the unknown class and the imbalanced dataset issues, as well as to improve the prediction accuracy. The presented system

[3]https://challenge2019.isic-archive.com/live-leaderboard.html

is 2nd from the top among groups not using external data, at the ISIC competition, at the time of the submission.

5. Plants Identification

5.1 Plant Identification with Large Number of Classes: SabanciU-GebzeTU System in PlantCLEF 2017

We describe the plant identification system that was submitted to the LifeCLEF plant identification campaign in 2017 [147], as a collaboration of Sabanci University and Gebze Technical University. Similar to our system that got a very close second place in 2016, we fine-tuned two well-known deep learning architectures (VGGNet and GoogLeNet) that were pre-trained on the object recognition dataset of ILSVRC 2012 and used an ensemble of 4-9 networks using score-level combination. Our best system was obtained with a classifier fusion of 9 networks trained with some differences in train settings, achieving an average inverse rank of 0.634 on the official test data, while the first place system achieved an impressive score of 0.92.

5.1.1 Introduction

Automatic plant identification addresses the identification of the plant species in a given photograph. Plant identification challenge within the Conference and Labs of the Evaluation Forum (CLEF) [148, 149, 150, 151, 152, 153, 147] is the most well-known annual event that benchmarks content-based image retrieval of plants. The campaign has been run since 2011, with plant species and number of training images almost doubling every year, reaching to 10,000 classes in the 2017 evaluation. Considering very high similarities between species and a large variety of imaging and plant conditions, the problem is rather challenging.

Our team participated in the PlantCLEF 2017 campaign under the name of *SabanciU-GebzeTU*. In all of our runs, we used an ensemble of 4-9 convolutional

networks, with different classifier combination criteria. The base networks were pre-trained deep convolutional neural networks of GoogLeNet [154] and VGGNet [155] that were fine-tuned with plant images. The campaign organizers provided two separate data sets: the main training set consisted of 256,203 images with clean labels *Encyclopedia of Life (EOL)* and the web crawled data consisted of around 1.6 million of images with noisy labels. The test set was sequestered until a few weeks before results submission. Details of the campaign can be found in [147].

The rest of this paper is organized as follows. 5.1.2 describes our approach based on: fine-tuning GoogLeNet and VGGNet models for plant identification and applying score-level classifier fusion. 5.1.3 describes the data sets and experimental results. The paper concludes in 5.1.4 with the summary and discussion of the utilized methods and obtained results.

5.1.2 Approach

Our approach was fine-tuning and fusing of two successful deep learning models, i.e. GoogLeNet [154] and VGGNet[155], using the implementations provided in the Caffe deep learning framework [156]. These models are, respectively, the first-ranked and second-ranked architectures of the ImageNet Large-Scale Visual Recognition Challenge (ILSVRC) 2014–both trained on the ILSVRC 2012 dataset with 1.2 million labeled images of 1,000 object classes.

In this work, we fine-tuned the GoogLeNet and VGGNet models starting from the learned weights of our PlantCLEF2016 system [157]. In the first network, we used only the training portion of EOL with internal augmentation (during training at each iteration a random crop of the image is used and randomly mirrored horizontally), to get some quick results. This network was the VGGNet architecture with all but the last layer of weights being fixed. In fact, in all of the experiments, we could only fine-tune the last 1-2 layers, as learning was very slow otherwise. This network achieved 41% accuracy.

After getting the base system running, we started using 8-fold external augmentation for training and later we started to incorporate images from the noisy dataset into the training data: as the web crawled data is not reliable, we tested 200,000 images from the noisy data set using the best networks we had thus far and took only those images for which prediction matched the groundtruth.

We also tried VGGNET using Batch Normalization and GoogleNet architecture, with roughly similar performance. In both of these networks, all of the layers were

fixed except for the last one due to scarce computing resources. Another network concentrated on the most common 1000 species and while we found that this network only achieved a 27% accuracy, it helped improve the performance of the ensemble like all other networks. In this fashion, each successive network (for a total of 9 different ones) was trained for either more iterations, or with new data added, or with different network architecture. At last, we trained one of the previous networks with *all* available training data, merging the validation set to the training set. This was done for only one network given the limited time.

Score-level averaging is applied to combine the prediction scores assigned to each of the augmented patches within a single network. As for the final systems, the obtained scores from all networks are combined using Borda count [158] or based on the maximum score of different classifiers.

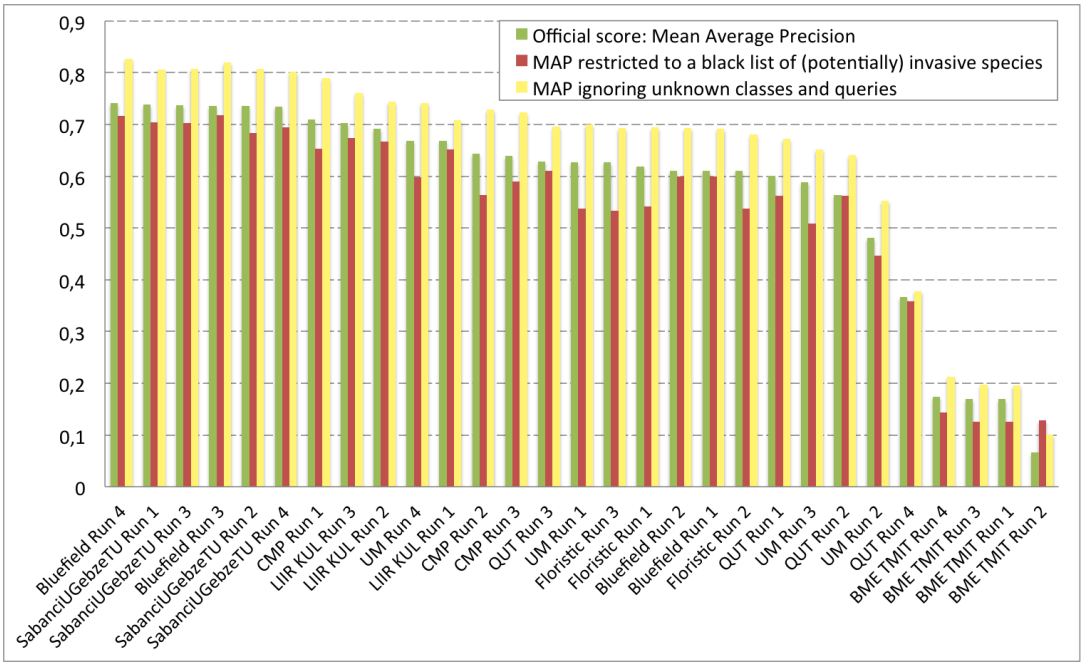


Figure 5.1 The official released results of PlantCLEF 2017

Our main problem was computational resources, faced with a very large number of classes and large amount of data. All trains and tests were run on a Linux system with a Tesla K40c and 12GB of video memory and in most cases training a network took 2-3 days.

5.1.3 Experimental Results

For training and validating our system, we used the EOL data consisting of 256,203 images of different plant organs, belonging to 10,000 species. Specifically, we ran-

Table 5.1 Rank comparison of the CLEF2017 published results that used (EOL) and (EOL+Noisy) data set

	Trusted (EOL)		
Run	Score	Top 1	Top 5
CMP Run 3	0.807	0.741	0.887
FHDO_BCSG Run 1	0.792	0.723	0.878
KDETUT Run 1	0.772	0.707	0.85
CMP Run 4	0.733	0.641	0.849
UM Run 1	0.7	0.621	0.795
PlantNet Run 1	0.613	0.513	0.734
SabanciUGebzeTU Run 2	0.581	0.508	0.68
	Trusted (EOL) + Noisy		
Run	Score	Top 1	Top 5
MarioTsaBerlin Run 4	0.92	0.885	0.962
KDETUT Run 4	0.853	0.793	0.927
KDETUT Run 3	0.837	0.769	0.922
UM Run 3	0.798	0.727	0.886
UM Run 4	0.789	0.715	0.882
SabanciUGebzeTU Run 4	0.638	0.557	0.738
SabanciUGebzeTU Run 1	0.636	0.556	0.737
SabanciUGebzeTU Run 3	0.622	0.537	0.728

domly divided the training portion of the dataset into two subsets for training and validation, with 174,280 and 81,923 images, respectively. The test portion of the dataset consists of a separate set of 25,170 images that was sequestered by the organizers, until the last weeks of the campaign. We will call these three subsets train, validation and test subsets respectively in the remainder of this paper. The base accuracy of the networks trained with all of the 10,000 classes ranged from 41% to 48.4% and the combined accuracy was 61.03%, on the validation subset. The combination was helpful even with highly correlated networks and taking less successful networks from the ensemble always reduced the performance The most successful network, based on the accuracy of the validation set, was the VGGNet using the largest training set (the train subset and around 60,000 samples from noisy data) and with a large batch size (60). The submitted runs are described below and the

results (mean inverse rank) released by the campaign organizers are shown in Figure 5.1 and given in [147].

More details of the used training set in our experiments with rank comparison are shown in 5.1.

- Run 1. In this run, the combination was done based on Borda count, with classifier confidence to break the ties.
- Run 2. This ensemble only used based systems trained with EOL data.
- Run 3. This system was the same as System 4 except for using a combination based on maximum confidence.
- Run 4. This system was the same as System 1 except for classifier combination weights.

5.1.4 Conclusions

The main objective was to preserve the high scores we obtained in 2016, despite the 10-fold increase in the number of classes [157]. Unfortunately, the large number of classes and limited computational power made it impossible to successfully fine-tune the networks. While our results were significantly below the best performing system this year, our results are not too far from our results last year, despite 10-fold increase in classes. It was also a challenging exercise to deal with a large, real life problem.

5.2 Plant Identification with Deep Learning Ensembles in ExpertLife-CLEF 2018

This work describes the plant identification system that we submitted to the ExpertLifeCLEF plant identification campaign in 2018. We fine-tuned two pre-trained deep learning architectures (SeNet and DensNetwork) using images shared by the CLEF organizers in 2017. Our main runs are 4 ensembles obtained with different weighted combinations of the 4 deep learning architectures. The fifth ensemble is based on deep learning features but uses Error Correcting Output Codes (ECOC) as the ensemble. Our best system has achieved a classification accuracy of 74.4%, while the best system obtained 86.7% accuracy, on the whole of the official test data. This system ranked 4th place among all the teams, but matched the accuracy of one of the human experts.

5.2.1 Introduction

Automatic plant identification is the problem of identifying the given plant species in a given photograph. Plant identification challenge of the Conference and Labs of the Evaluation Forum (CLEF) [148, 149, 150, 151, 152, 153, 159, 160] is the most well-known annual event that benchmarks the progress in identification of plant species. The campaign has been running since 2011, with plant species reaching 10,000 classes in the 2017 evaluation.

The emphasis of the campaign changes slightly from year to year, while the core of the campaign is to benchmark plant identification progress. This year's emphasis was on measuring automatic systems' performances with that of human experts. For that reason, a subset of the test data was labelled by human experts and the systems were evaluated on their accuracy on the whole test set, as well as their performance on the subset. The details of the plant identification and the overall LifeCLEF campaigns are described in [160] and [161] respectively.

We have been participating into this campaign since 2011, first with traditional approaches and carefully selected features [162, 163, 164] and then with deep learning approaches [157]. While the traditional approaches worked well on the simpler problem of leaf based identification (leaf images on simple backgrounds), deep learning approaches brought a significant increase in accuracy despite much increased problem complexity (unrestricted photographs and 10,000 classes).

This year our team participated in the ExpertLifeCLEF2018 challenge under the name of *SabanciU-GTU*. In our main 4 runs (Runs 1, 3, 4, 5), we have used an ensemble of four convolutional networks according to different combination weights. The networks were pre-trained deep convolutional neural networks of SeNet [165] and DensNetwork [166] that were fine-tuned with plant images. In the fifth system, we took the deep learning features (last convolutional layer activations) of our SeNet system and trained 200 different binary classifiers to form an Error Correcting Codes (ECOC) ensemble.

The training data was obtained from CLEF, as a combination of data collected from the Encyclopedia of Life (EOL) and images collected from the web and shared by CLEF in 2017. This latter set is noisy as it is not verified by experts for correctness. The submitted systems were different combination schemes applied to the four models.

The rest of this paper is organized as follows. 5.2.2 describes the proposed methods based on the fine-tuning of SeNet and DensNetwork models for plant identification, data augmentation, and classifiers' fusion. 5.2.4 is dedicated to the description of the utilized dataset and presentation of designed experiments and their results. The paper concludes in 5.2.5 with the summary and discussion of the utilized methods and obtained results.

5.2.2 Core System

Our approach was based on fine-tuning and fusing of two successful deep learning models, namely SeNet [165] and DensNetwork[166]. These models are, respectively, the first-ranked and second-ranked architectures of the ImageNet Large-Scale Visual Recognition Challenge (ILSVRC) 2014–both trained on the ILSVRC 2012 dataset with 1.2 million labeled images of 1,000 object classes.

SeNet [165], Winner of ImageNet 2017 Classification Task [140], introduces a building block for convolution neural networks that improves channel inter-dependencies. The main idea is to weight each channel adaptively based on its importance. SE-block is flexible which means that it can be integrated into any modern deep learning architecture. In this work, we utilized SE-blocks with ResNet-50 [51] module.

DensNetwork [166] are built from dense blocks and pooling operations where there is a connection between each block to every receding blocks. Thus, with n blocks, there are $n(n+1)/2$ direct connections. Input of each dense block is an iterative concatenation of previous feature maps. One of the advantages of DensNetwork is

that it lessens the vanishing-gradient problem which makes it easy to train.

Score-level averaging is applied to combine the prediction scores assigned to each class for all the augmented patches within a single network and then for combining the scores obtained for different images of the same unique plant (called an "observation" in the campaign terminology).

All training and tests were run on a linux system with a Titan X Pascal GPU and 12GB of video memory.

5.2.3 Error-Correcting Output Codes

As a second ensemble approach, we tried the Error Correcting Output Codes (ECOC) approach [93]. In ECOC, a number of binary classifiers are trained such that each one is assigned a separate dichotomy of the classes, which is defined by a given ECOC matrix. In the ECOC matrix M, the jth column indicates the dichotomy assigned for base classifier h_j. That is, a particular element $M_{ij}\epsilon\{+1, -1\}$ indicates the desired label for class c_i to be used in training the base classifier h_j. The ith row of M, denoted as M_i, is the codeword for class c_i indicating the desired output for that class.

A given test instance x is first classified by each base classifier, obtaining the output vector $y = [y_1, ..., y_L]$ where y_j is the output of the classifier h_j for the given input x. Then, the distance between y and the codeword M_i of class c_i is computed by using a distance metric such as the Hamming distance. The class c_k for which this distance is minimum, is chosen as the estimated class label:

$$k = argmin_{i=1...K}\ d(y, M_i)$$

We took the deep learning features (last convolutional layer activations) of our SeNet system (System2) and trained 200 different binary classifiers according to the predetermined ECOC matrix.

5.2.4 Experiments and Results

The first three systems are trained using SeNet-ResNet-50 architecture. For training the first system, we only used the EOL data consisting of $256,203$ images of differ-

ent plant organs, belonging to 10,000 species. Internal augmentation was applied during training (at each iteration, a random crop of the image is used and randomly mirrored horizontally). For validation, we used the plant test dataset of LifeCLEF 2017 consisting of 25,170 images.

For the second system, several data augmentation are applied to the training images like saliency detection [167], flip, and several rotation angles. In total, number of images in the training dataset after augmentation is around 4,500,000 images and the system was trained over 10 epochs. For the third system, we trained using all of the available data with augmentation (EOL data, web-collected noisy data, and testing set of LifeCLEF2017) excluding 1,000 images from test 2017 for validation. This system was trained over 25 epochs. The fourth system is trained using DensNetwork using the same training data as in System 3. Training DensNetwork was quite slow, therefore, we trained system 4 over only 5 epochs.

We implemented SeNet and DensNetwork models using the Caffe deep learning framework [156]. All the weights were fine-tuned, while the last layer was learned from scratch. We used the same learning rate for all of the system which is 0.01.

- **Run 1,3,4,5**
  Different weighted combinations of the same basic four deep learning systems described in Section 5.2.2. In System 5, that was the best performing system by a 0.001 margin, we used the image quality information that is given inside the metadata in the xml files. The score of each image is weighted using the quality information. In the absence of quality information, no weighting is applied.

- **Run 2** The ECOC ensemble where 200 base classifiers were trained on binary classification tasks set forth according to a predetermined, random ECOC matrix. The ECOC matrix was initialized randomly, and then simulated annealing was used to increase the Hamming distance between rows. As features, we used the deep learning features obtained from the last convolutional layer of first system described above, and trained 2-hidden layer shallow networks (500 hidden nodes at each layer) as base classifiers.

  While the accuracy of this system fell short of the performance of the deep learning architectures, the system shows promise in that the accuracy increases as we increase the number of base classifiers: from 51% with 100 base classifiers, to 59% and 61% with 200 and 300 base classifiers, on the LifeCLEF 2017 test data. The training times are also less than one tenth of that of one deep architecture (around 2-3 hours per 100 base classifiers on an iMac).

As a promising and fast alternative, we are planning on working on improvements of the ECOC ensemble as proposed in [112] and [168].

5.2.4.1 Test Results.

We submitted the classification results of the before mentioned systems on the official test set of the ExpertLifeCLEF 2018. The utilized official metric for evaluation was the average accuracy on a small subset of the test data that was also identified by human experts. Results on the whole test set were also provided. The released results by the challenge organizers are shown in 5.2 and given in [161].

Our best system has achieved a top-1 classification accuracy of 74.4%, while the best system obtained 86.7% accuracy on the whole official test data. This system ranked 4th place among all the teams, but matched the accuracy of one of the human experts.

Our results for the small subset that is also labelled by human experts is 61.3%, while the 9 human experts scores range from 96% to 61.3%, on this subset. In other words, our best system has reached the top-1 identification accuracy of one of the human experts.

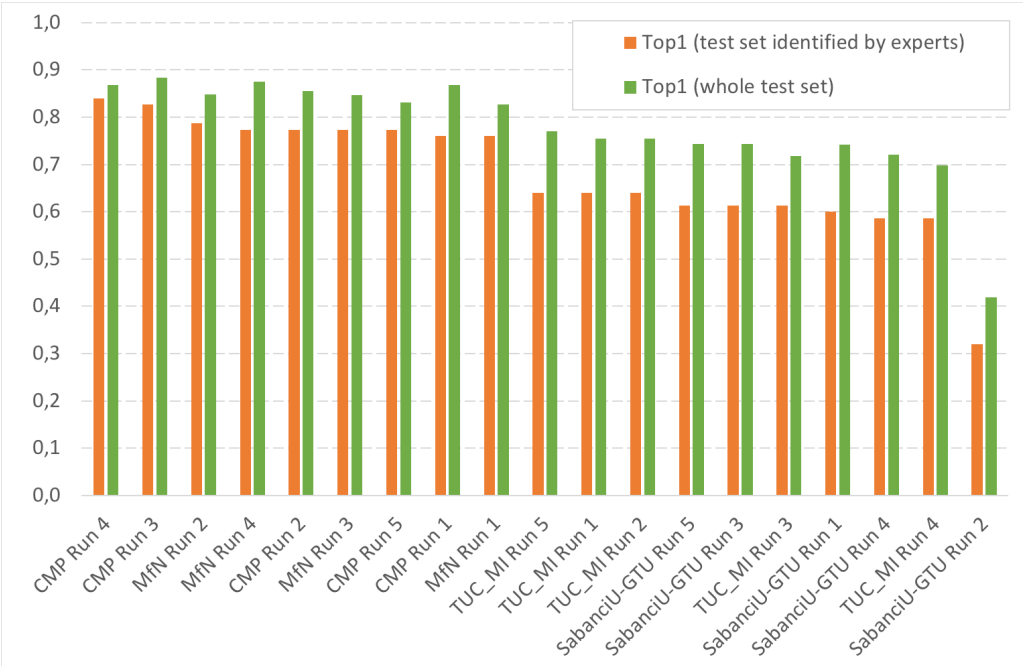


Figure 5.2 The official released results of ExpertLifeCLEF 2018

5.2.5 Conclusions

The competition that has been running for several years now has seen a shift from hand-crafted features and to deep learning classifiers in the last years. Our goal this year was to use the best performing pre-trained architectures while diversifying the base classifiers within the ensemble. Considering the fact that we only had one machine with GPU, we consider the performance of our system (74.4% accuracy) satisfactory on such a complex problem (10,000 classes). For the future, we plan to work on better ensemble techniques with deep architectures, including improvements of the ECOC ensemble.

5.2.5.1 Acknowledgments.

We gratefully acknowledge NVIDIA Corporation with the donation of the Titan X Pascal GPU used in this research.

Bibliography

[1] Matthew R Boutell, Jiebo Luo, Xipeng Shen, and Christopher M Brown. Learning multi-label scene classification. *Pattern Recognition*, 37(9):1757–1771, 2004.

[2] Mahdi M Kalayeh, Boqing Gong, and Mubarak Shah. Improving facial attribute prediction using semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6942–6950, 2017.

[3] Constantine P Papageorgiou, Michael Oren, and Tomaso Poggio. A general framework for object detection. In *Sixth International Conference on Computer Vision (IEEE Cat. No. 98CH36271)*, pages 555–562. IEEE, 1998.

[4] Antitza Dantcheva, Petros Elia, and Arun Ross. What else does your biometric data reveal? a survey on soft biometrics. *IEEE Transactions on Information Forensics and Security*, 11(3):441–467, 2016.

[5] Ohil K Manyam, Neeraj Kumar, Peter Belhumeur, and David Kriegman. Two faces are better than one: Face recognition in group photographs. In *Biometrics (IJCB), 2011 Int. Joint Conf. on*, pages 1–8. IEEE, 2011.

[6] Thomas Berg and Peter N Belhumeur. Poof: Part-based one-vs.-one features for fine-grained categorization, face verification, and attribute estimation. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 955–962, 2013.

[7] Neeraj Kumar, Alexander C Berg, Peter N Belhumeur, and Shree K Nayar. Attribute and simile classifiers for face verification. In *IEEE 12th International Conference on Computer Vision (ICCV)*, pages 365–372. IEEE, 2009.

[8] Christoph H Lampert, Hannes Nickisch, and Stefan Harmeling. Attribute-based classification for zero-shot visual object categorization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(3):453–465, 2014.

[9] Lubomir Bourdev, Subhransu Maji, and Jitendra Malik. Describing people: A poselet-based approach to attribute classification. In *International Conference on Computer Vision (ICCV)*, pages 1543–1550. IEEE, 2011.

[10] Yan Li, Ruiping Wang, Haomiao Liu, Huajie Jiang, Shiguang Shan, and Xilin Chen. Two birds, one stone: Jointly learning binary code for large-scale face image retrieval and attributes prediction. In *IEEE International Conference on Computer Vision (ICCV)*, pages 3819–3827, 2015.

[11] Ali Sharif Razavian, Hossein Azizpour, Josephine Sullivan, and Stefan Carlsson. Cnn features off-the-shelf: an astounding baseline for recognition. In *Proc. of the IEEE Conf. on CVPR workshops*, pages 806–813, 2014.

[12] Fengyi Song, Xiaoyang Tan, and Songcan Chen. Exploiting relationship between attributes for improved face verification. *Computer Vision and Image Understanding*, 122:143–154, 2014.

[13] Zhenyao Zhu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Multi-view perceptron: a deep model for learning face identity and view representations. In *NIPS*, pages 217–225, 2014.

[14] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *IEEE International Conference on Computer Vision (ICCV)*, pages 3730–3738, 2015.

[15] Andras Rozsa, Manuel Günther, Ethan M Rudd, and Terrance E Boult. Are facial attributes adversarially robust? In *23rd International Conference on Pattern Recognition (ICPR)*, pages 3121–3127. IEEE, 2016.

[16] Yang Zhong, Josephine Sullivan, and Haibo Li. Face attribute prediction using off-the-shelf CNN features. In *International Conference on Biometrics (ICB)*, pages 1–7. IEEE, 2016.

[17] Sara Atito Aly and Berrin Yanikoglu. Multi-label networks for face attributes classification. In *2018 IEEE International Conference on Multimedia & Expo Workshops (ICMEW)*, pages 1–6. IEEE, 2018.

[18] Sara Atito Ali Ahmed and Berrin Yanikoglu. Within-network ensemble for face attributes classification. In *International Conference on Image Analysis and Processing*, pages 466–476. Springer, 2019.

[19] Sara Atito Ali Ahmed and Berrin Yanikoglu. Relative attribute classification with deep-ranksvm. In *Pattern Recognition. ICPR International Workshops and Challenges: Virtual Event, January 10–15, 2021, Proceedings, Part II*, pages 659–671. Springer, 2021.

[20] Josef Kittler, Mohamad Hater, and Robert PW Duin. Combining classifiers. In *Proceedings of 13th international conference on pattern recognition*, volume 2, pages 897–901. IEEE, 1996.

[21] Josef Kittler. Combining classifiers: A theoretical framework. *Pattern analysis and Applications*, 1(1):18–27, 1998.

[22] Lionel S Penrose. The elementary statistics of majority voting. *Journal of the Royal Statistical Society*, 109(1):53–57, 1946.

[23] Venkatesan Guruswami and Amit Sahai. Multiclass learning, boosting, and error-correcting codes. In *Proceedings of the twelfth annual conference on Computational learning theory*, pages 145–155, 1999.

[24] Saso Džeroski and Bernard Ženko. Is combining classifiers with stacking better than selecting the best one? *Machine learning*, 54(3):255–273, 2004.

[25] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016.

[26] Yawen Xiao, Jun Wu, Zongli Lin, and Xiaodong Zhao. A deep learning-based multi-model ensemble method for cancer prediction. *Computer methods and programs in biomedicine*, 153:1–9, 2018.

[27] Nils Gessert, Maximilian Nielsen, Mohsin Shaikh, René Werner, and Alexander Schlaefer. Skin lesion classification using ensembles of multi-resolution efficientnets with meta data. *MethodsX*, page 100864, 2020.

[28] Xudie Ren, Haonan Guo, Shenghong Li, Shilin Wang, and Jianhua Li. A novel image classification method with cnn-xgboost model. In *International Workshop on Digital Watermarking*, pages 378–390. Springer, 2017.

[29] Long Pang, Junjie Wang, Lingling Zhao, Chunyu Wang, and Hui Zhan. A novel protein subcellular localization method with cnn-xgboost model for alzheimer's disease. *Frontiers in genetics*, 9:751, 2019.

[30] Yun Ju, Guangyu Sun, Quanhe Chen, Min Zhang, Huixian Zhu, and Mujeeb Ur Rehman. A model combining convolutional neural network and lightgbm algorithm for ultra-short-term wind power forecasting. *IEEE Access*, 7:28309–28318, 2019.

[31] Sara Atito Ali Ahmed, Cemre Zor, Muhammad Awais, Berrin Yanikoglu, and Josef Kittler. Deep convolutional neural network ensembles using ecoc. *IEEE Access*, 9:86083–86095, 2021.

[32] Sara Atito Ali Ahmed, Berrin Yanikoğlu, Özgü Göksu, and Erchan Aptoula. Skin lesion classification with deep cnn ensembles. In *2020 28th Signal Processing and Communications Applications Conference (SIU)*, pages 1–4. IEEE, 2020.

[33] Sara Atito Ali Ahmed, Berrin Yanikoglu, Cemre Zor, Muhammad Awais, and Josef Kittler. Skin lesion diagnosis with imbalanced ecoc ensembles. In *International Conference on Machine Learning, Optimization, and Data Science*, pages 292–303. Springer, 2020.

[34] Sara Atito Ali Ahmed, Berrin Yanıkoğlu, and Erchan Aptoula. Plant identification with large number of classes: Sabanciu-gebzetu system in plantclef 2017. CLEF, 2017.

[35] Sara Atito, Berrin A Yanikoglu, Erchan Aptoula, Ipek Ganiyusufoglu, Aras Yildiz, Kerem Yildirir, Baris Sevilmis, and M Umut Sen. Plant identification with deep learning ensembles. In *CLEF (Working Notes)*, 2018.

[36] Rogerio Schmidt Feris, Christoph Lampert, and Devi Parikh. *Visual Attributes*. Springer, 2017.

[37] Emily M Hand and Rama Chellappa. Attributes for improved attributes: A multi-task network utilizing implicit and explicit relationships for facial attribute classification. In *AAAI*, pages 4068–4074, 2017.

[38] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv:1409.1556*, 2014.

[39] Rajeev Ranjan, Vishal M Patel, and Rama Chellappa. Hyperface: A deep multi-task learning framework for face detection, landmark localization, pose estimation, and gender recognition. *arXiv:1603.01249*, 2016.

[40] Sihua Yi, Nan Jiang, Bin Feng, Xinggang Wang, and Wenyu Liu. Online similarity learning for visual tracking. *Information Sciences*, 364:33–50, 2016.

[41] Wenhao Huang, Guojie Song, Haikun Hong, and Kunqing Xie. Deep architecture for traffic flow prediction: deep belief networks with multitask learning. *IEEE Transactions on Intelligent Transportation Systems*, 15(5):2191–2201, 2014.

[42] Yong Luo, Dacheng Tao, Bo Geng, Chao Xu, and Stephen J Maybank. Manifold regularized multitask learning for semi-supervised multilabel image classification. *IEEE Transactions on Image Processing*, 22(2):523–536, 2013.

[43] Jonathan Baxter. A bayesian/information theoretic model of learning to learn via multiple task sampling. *Machine Learning*, 28(1):7–39, 1997.

[44] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional architecture for fast feature embedding. In *Proc. of the 22nd Int. Conf. on Multimedia*, pages 675–678. ACM, 2014.

[45] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. Show, attend and tell: Neural image caption generation with visual attention. In *International Conference on Machine Learning*, pages 2048–2057, 2015.

[46] Gary B Huang, Manu Ramesh, Tamara Berg, and Erik Learned-Miller. Labeled faces in the wild: A database for studying face recognition in unconstrained environments. Technical report, 07-49, University of Massachusetts, Amherst, Technical Report, October, 2007.

[47] Hu Han, Anil K Jain, Fang Wang, Shiguang Shan, and Xilin Chen. Heterogeneous face attribute estimation: A deep multi-task learning approach. *IEEE transactions on pattern analysis and machine intelligence*, 40(11):2597–2609, 2017.

[48] Max Ehrlich, Timothy J Shields, Timur Almaev, and Mohamed R Amer. Facial attributes classification using multi-task representation learning. In *IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 47–55, 2016.

[49] Ethan M Rudd, Manuel Günther, and Terrance E Boult. Moon: A mixed objective optimization network for the recognition of facial attributes. In *European Conference on Computer Vision (ECCV)*, pages 19–35. Springer, 2016.

[50] Hu Han, Anil K Jain, Fang Wang, Shiguang Shan, and Xilin Chen. Heterogeneous face attribute estimation: A deep multi-task learning approach. *IEEE transactions on Pattern Analysis and Machine Intelligence*, 40(11):2597–2609, 2018.

[51] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *IEEE conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

[52] Ekin D Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, and Quoc V Le. Autoaugment: Learning augmentation policies from data. *arXiv:1805.09501*, 2018.

[53] Adriana Kovashka, Devi Parikh, and Kristen Grauman. Whittlesearch: Interactive image search with relative attribute feedback. *International Journal of Computer Vision*, 115(2):185–210, 2015.

[54] Adriana Kovashka and Kristen Grauman. Attributes for image retrieval. In *Visual Attributes*, pages 89–117. Springer, 2017.

[55] Lin Chen, Peng Zhang, and Baoxin Li. Instructive video retrieval based on hybrid ranking and attribute learning: A case study on surgical skill training. In *Proceedings of the 22nd ACM*, pages 1045–1048, 2014.

[56] Yingwei Pan, Ting Yao, Houqiang Li, and Tao Mei. Video captioning with transferred semantic attributes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6504–6512, 2017.

[57] Ting Yao, Yingwei Pan, Yehao Li, Zhaofan Qiu, and Tao Mei. Boosting image captioning with attributes. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4894–4902, 2017.

[58] Yanwei Fu, Tao Xiang, Yu-Gang Jiang, Xiangyang Xue, Leonid Sigal, and Shaogang Gong. Recent advances in zero-shot recognition: Toward data-efficient understanding of visual content. *IEEE Signal Processing Magazine*, 35(1):112–125, 2018.

[59] Ankan Bansal, Karan Sikka, Gaurav Sharma, Rama Chellappa, and Ajay Divakaran. Zero-shot object detection. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 384–400, 2018.

[60] Ni Zhuang, Yan Yan, Si Chen, Hanzi Wang, and Chunhua Shen. Multi-label learning based deep transfer neural network for facial attribute classification. *Pattern Recognition*, 80:225–240, 2018.

[61] Devi Parikh and Kristen Grauman. Relative attributes. In *International Conference on Computer Vision*, pages 503–510. IEEE, 2011.

[62] Yaser Souri, Erfan Noury, and Ehsan Adeli. Deep relative attributes. In *Asian conference on computer vision*, pages 118–133. Springer, 2016.

[63] Shaoxin Li, Shiguang Shan, and Xilin Chen. Relative forest for attribute prediction. In *Asian Conference on Computer Vision*, pages 316–327. Springer, 2012.

[64] Aron Yu and Kristen Grauman. Fine-grained visual comparisons with local learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 192–199, 2014.

[65] Fanyi Xiao and Yong Jae Lee. Discovering the spatial extent of relative attributes. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1458–1466, 2015.

[66] Aron Yu and Kristen Grauman. Just noticeable differences in visual attributes. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2416–2424, 2015.

[67] Krishna Kumar Singh and Yong Jae Lee. End-to-end localization and ranking for relative attributes. In *European Conference on Computer Vision*, pages 753–769. Springer, 2016.

[68] Xiaoshan Yang, Tianzhu Zhang, Changsheng Xu, Shuicheng Yan, M Shamim Hossain, and Ahmed Ghoneim. Deep relative attributes. *IEEE Transactions on Multimedia*, 18(9):1832–1842, 2016.

[69] Zeshang Zhang, Yingming Li, and Zhongfei Zhang. Relative attribute learning with deep attentive cross-image representation. In *Asian Conference on Machine Learning*, pages 879–892, 2018.

[70] Aude Oliva and Antonio Torralba. Modeling the shape of the scene: A holistic representation of the spatial envelope. *International journal of computer vision*, 42(3):145–175, 2001.

[71] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012.

[72] Thorsten Joachims. Optimizing search engines using clickthrough data. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '02, page 133–142, New York, NY, USA, 2002. Association for Computing Machinery.

[73] Lukas Ruff, Robert Vandermeulen, Nico Goernitz, Lucas Deecke, Shoaib Ahmed Siddiqui, Alexander Binder, Emmanuel Müller, and Marius Kloft. Deep one-class classification. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 4393–4402, Stockholmsmässan, Stockholm Sweden, 10–15 Jul 2018. PMLR.

[74] Ramachandruni N Sandeep, Yashaswi Verma, and CV Jawahar. Relative parts: Distinctive parts for learning relative attributes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3614–3621, 2014.

[75] Gary B. Huang, Manu Ramesh, Tamara Berg, and Erik Learned-Miller. Labeled faces in the wild: A database for studying face recognition in unconstrained environments. Technical Report 07-49, University of Massachusetts, Amherst, October 2007.

[76] Aron Yu and Kristen Grauman. Semantic jitter: Dense supervision for visual comparisons via synthetic images. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 5570–5579, 2017.

[77] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alexander A Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[78] Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V Le. Learning transferable architectures for scalable image recognition. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 8697–8710, 2018.

[79] Leo Breiman. Bagging predictors. *Machine learning*, 24(2):123–140, 1996.

[80] Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[81] Eun Bae Kong and Thomas G Dietterich. Error-correcting output coding corrects bias and variance. In *Machine Learning Proceedings 1995*, pages 313–321. Elsevier, 1995.

[82] Yaniv Taigman, Ming Yang, Marc'Aurelio Ranzato, and Lior Wolf. Deepface: Closing the gap to human-level performance in face verification. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014.

[83] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.

[84] Pinkesh Badjatiya, Shashank Gupta, Manish Gupta, and Vasudeva Varma. Deep learning for hate speech detection in tweets. In *Proceedings of the 26th International Conference on World Wide Web Companion*, pages 759–760, 2017.

[85] Mario Lasseck. Image-based plant species identification with deep convolutional neural networks. In *CLEF (Working Notes)*, 2017.

[86] Mingxing Tan and Quoc V Le. Efficientnet: Rethinking model scaling for convolutional neural networks. *arXiv preprint arXiv:1905.11946*, 2019.

[87] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pages 785–794, 2016.

[88] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. Lightgbm: A highly efficient gradient boosting decision tree. In *Advances in neural information processing systems*, pages 3146–3154, 2017.

[89] Liudmila Prokhorenkova, Gleb Gusev, Aleksandr Vorobev, Anna Veronika Dorogush, and Andrey Gulin. Catboost: unbiased boosting with categorical features. In *Advances in neural information processing systems*, pages 6638–6648, 2018.

[90] Alberto Torres-Barrán, Álvaro Alonso, and José R Dorronsoro. Regression tree ensembles for wind energy and solar radiation prediction. *Neurocomputing*, 326:151–160, 2019.

[91] Alexis Hocquenghem. Codes correcteurs d'erreurs. *Chiffres*, 2(2):147–56, 1959.

[92] Raj Chandra Bose and Dwijendra K Ray-Chaudhuri. On a class of error correcting binary group codes. *Information and control*, 3(1):68–79, 1960.

[93] Thomas G. Dietterich and Ghulum Bakiri. Solving multiclass learning problems via error-correcting output codes. *Journal of Artificial Intelligence Research*, 2:263–286, 1995.

[94] Oriol Pujol, Petia Radeva, and Jordi Vitria. Discriminant ecoc: A heuristic method for application dependent design of error correcting output codes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(6):1007–1012, 2006.

[95] Miguel Ángel Bautista, Sergio Escalera, Xavier Baró, Petia Radeva, Jordi Vitrià, and Oriol Pujol. Minimal design of error-correcting output codes. *Pattern Recognition Letters*, 33(6):693–702, 2012.

[96] Gareth James and Trevor Hastie. The error coding method and picts. *Journal of Computational and Graphical statistics*, 7(3):377–387, 1998.

[97] Gareth James. *Majority vote classifiers: theory and applications*. PhD thesis, Stanford University, 1998.

[98] Sergio Escalera, Oriol Pujol, and Petia Radeva. On the decoding process in ternary error-correcting output codes. *IEEE Trans. Pattern Anal. Mach. Intell.*, 32:120–134, January 2010.

[99] Liu Xiao-Feng, Zhang Xue-ying, and Duan Ji-Kang. Speech recognition based on support vector machine and error correcting output codes. In *2010 First International Conference on Pervasive Computing, Signal Processing and Applications*, pages 336–339. IEEE, 2010.

[100] Qixiang Ye, Jixiang Liang, and Jianbin Jiao. Pedestrian detection in video images via error correcting output code classification of manifold subclasses.

IEEE Transactions on Intelligent Transportation Systems, 13(1):193–202, 2011.

[101] Raymond S Smith and Terry Windeatt. Facial action unit recognition using multi-class classification. *Neurocomputing*, 150:440–448, 2015.

[102] Shilin Gu, Yang Cai, Jincheng Shan, and Chenping Hou. Active learning with error-correcting output codes. *Neurocomputing*, 364:182–191, 2019.

[103] Bowen Zhang, Benedetta Tondi, and Mauro Barni. On the adversarial robustness of dnns based on error correcting output codes. *arXiv preprint arXiv:2003.11855*, 2020.

[104] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009.

[105] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng. Reading digits in natural images with unsupervised feature learning. 2011.

[106] David Hughes, Marcel Salathé, et al. An open access repository of images on plant health to enable the development of mobile disease diagnostics. *arXiv preprint arXiv:1511.08060*, 2015.

[107] Bradley Efron. Bootstrap methods: another look at the jackknife. In *Breakthroughs in statistics*, pages 569–593. Springer, 1992.

[108] Robert E Schapire. The boosting approach to machine learning: An overview. In *Nonlinear estimation and classification*, pages 149–171. Springer, 2003.

[109] David H Wolpert. Stacked generalization. *Neural networks*, 5(2):241–259, 1992.

[110] Ana Carolina Lorena and André CPLF De Carvalho. Building binary-tree-based multiclass classifiers using separability measures. *Neurocomputing*, 73(16-18):2837–2845, 2010.

[111] E. Alpaydin and E. Mayoraz. Learning error-correcting output codes from data. In *Proceedings of the 9th International Conference on Artificial Neural Networks (ICANN 1999).*, volume 2, pages 743 –748, 1999.

[112] Cemre Zor, Berrin Yanikoglu, Terry Windeatt, and Ethem Alpaydin. FLIP-ECOC: a greedy optimization of the ECOC matrix. In *Proceedings of the 25th International Symposium on Computer and Information Sciences (ISCIS 2010)*, pages 149 – 154. Springer, 2010.

[113] Cemre Zor, Berrin A. Yanikoglu, Erinc Merdivan, Terry Windeatt, Josef Kittler, and Ethem Alpaydin. Beamecoc: A local search for the optimization of the ECOC matrix. In *23rd International Conference on Pattern Recognition, ICPR 2016, Cancún, Mexico, December 4-8, 2016*, pages 198–203, 2016.

[114] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks.

In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4510–4520, 2018.

[115] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826, 2016.

[116] François Chollet. Xception: Deep learning with depthwise separable convolutions. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 1251–1258, 2017.

[117] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7132–7141, 2018.

[118] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[119] Andrea Vedaldi and Karel Lenc. Matconvnet: Convolutional neural networks for matlab. In *Proceedings of the 23rd ACM international conference on Multimedia*, pages 689–692, 2015.

[120] Igor E Kuralenok, Yurii Rebryk, Ruslan Solovev, and Anton Ermilov. Factorized multiclass boosting. *arXiv preprint arXiv:1909.04904*, 2019.

[121] Sharada P Mohanty, David P Hughes, and Marcel Salathé. Using deep learning for image-based plant disease detection. *Frontiers in plant science*, 7:1419, 2016.

[122] Edna Chebet Too, Li Yujian, Sam Njuki, and Liu Yingchun. A comparative study of fine-tuning deep learning models for plant disease identification. *Computers and Electronics in Agriculture*, 161:272–279, 2019.

[123] KC Kamal, Zhendong Yin, Mingyang Wu, and Zhilu Wu. Depthwise separable convolution architectures for plant disease classification. *Computers and Electronics in Agriculture*, 165:104948, 2019.

[124] H. Kittler, H. Pehamberger, K. Wolff, and M. Binder. Diagnostic accuracy of dermoscopy. *The lancet oncology*, 3:159–165, 2002.

[125] A.B. Kimball and J.S. Resneck Jr. The US dermatology workforce: a specialty remains in shortage. *Journal of the American Academy of Dermatology*, 59:741–745, 2008.

[126] B Scholköpf and A Smola. Support vector machines, regularization, optimization, and beyond. *Learning with Kernels*, 2002.

[127] Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. Isolation forest. In *2008 Eighth IEEE International Conference on Data Mining*, pages 413–422. IEEE, 2008.

[128] Trevor Hastie and Robert Tibshirani. Discriminant analysis by gaussian mixtures. *Journal of the Royal Statistical Society: Series B (Methodological)*, 58(1):155–176, 1996.

[129] Philipp Tschandl, Cliff Rosendahl, and Harald Kittler. The ham10000 dataset, a large collection of multi-source dermatoscopic images of common pigmented skin lesions. *Scientific data*, 5:180161, 2018.

[130] M. Combalia, N. C. F. Codella, V. Rotemberg, B. Helba, V. Vilaplana, O. Reiter, A. C. Halpern, S. Puig, and J. Malvehy. Bcn20000: Dermoscopic lesions in the wild, 2019.

[131] N. C. F. Codella, D. Gutman, M. E. Celebi, B. Helba, M. A. Marchetti, S. W. Dusza, A. Kallo, K. Liopyris, N. Mishraand H. Kittler, et al. Skin lesion analysis toward melanoma detection. In *IEEE 15th International Symposium on Biomedical Imaging (ISBI)*, pages 168–172, 2018.

[132] Hiam Alquran, Isam Abu Qasmieh, Ali Mohammad Alqudah, Sajidah Alhammouri, Esraa Alawneh, Ammar Abughazaleh, and Firas Hasayen. The melanoma skin cancer detection and classification using support vector machine. In *2017 IEEE Jordan Conference on Applied Electrical Engineering and Computing Technologies (AEECT)*, pages 1–5. IEEE, 2017.

[133] Andre Esteva, Brett Kuprel, Roberto A Novoa, Justin Ko, Susan M Swetter, Helen M Blau, and Sebastian Thrun. Dermatologist-level classification of skin cancer with deep neural networks. *Nature*, 542(7639):115–118, 2017.

[134] David Gutman, Noel CF Codella, Emre Celebi, Brian Helba, Michael Marchetti, Nabin Mishra, and Allan Halpern. Skin lesion analysis toward melanoma detection: A challenge at the international symposium on biomedical imaging (isbi) 2016, hosted by the international skin imaging collaboration (isic). *arXiv preprint arXiv:1605.01397*, 2016.

[135] Thomas G Dietterich and Ghulum Bakiri. Solving multiclass learning problems via error-correcting output codes. *Journal of artificial intelligence research*, 2:263–286, 1994.

[136] Sara Atito, Berrin A. Yanikoglu, and Erchan Aptoula. Plant identification with large number of classes: Sabanciu-gebzetu system in plantclef 2017. In *Working Notes of CLEF 2017 - Conference and Labs of the Evaluation Forum, Dublin, Ireland, September 11-14, 2017*, volume 1866, 2017.

[137] Chuanqi Tan, Fuchun Sun, Tao Kong, Wenchang Zhang, Chao Yang, and Chunfang Liu. *A Survey on Deep Transfer Learning: 27th International Conference on Artificial Neural Networks, Rhodes, Greece, October 4–7, 2018, Proceedings, Part III*, pages 270–279. 2018.

[138] Ikram Ud Din, Joel Rodrigues, and Naveed Islam. A novel deep learning based framework for the detection and classification of breast cancer using transfer learning. *Pattern Recognition Letters*, 125, 2019.

[139] Sara Atito, Berrin A. Yanikoglu, Erchan Aptoula, Ipek Ganiyusufoglu, Aras Yildiz, Kerem Yildirir, Baris Sevilmis, and M. Umut Sen. Plant identification with deep learning ensembles. In *Working Notes of CLEF 2018 - Conference and Labs of the Evaluation Forum, Avignon, France, September 10-14, 2018*, volume 2125, 2018.

[140] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015.

[141] Michael J. Kearns and Leslie G. Valiant. Learning boolean formulae or finite automata is as hard as factoring. 1988.

[142] Cemre Zor, Terry Windeatt, and Berrin A. Yanikoglu. Bias-variance analysis of ecoc and bagging using neural nets. In *Ensembles in Machine Learning Applications*, 2011.

[143] Mingxia Liu, Daoqiang Zhang, Songcan Chen, and Hui Xue. Joint binary classifier learning for ecoc-based multi-class classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38:2335–2341, 2016.

[144] Jie Qin, Li Liu, Ling Shao, Fumin Shen, Bingbing Ni, Jiaxin Chen, and Yunhong Wang. Zero-shot action recognition with error-correcting output codes. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[145] Graham D Finlayson and Elisabetta Trezzi. Shades of gray and colour constancy. In *Color and Imaging Conference*, volume 2004, pages 37–41. Society for Imaging Science and Technology, 2004.

[146] Catarina Barata, M Emre Celebi, and Jorge S Marques. Improving dermoscopy image classification using color constancy. *IEEE journal of biomedical and health informatics*, 19(3):1146–1152, 2014.

[147] Alexis Joly, Hervé Goëau, Hervé Glotin, Concetto Spampinato, Pierre Bonnet, Willem-Pier Vellinga, Jean-Christophe Lombardo, Robert Planqué, Simone Palazzo, and Henning Müller. Lifeclef 2017 lab overview: multimedia species identification challenges. In *CLEF 2017 Proceedings, Springer Lecture Notes in Computer Science (LNCS)*, 2017.

[148] Hervé Goëau, Pierre Bonnet, Alexis Joly, Nozha Boujemaa, Daniel Barthelemy, Jean-François Molino, Philippe Birnbaum, Elise Mouysset, and Marie Picard. The CLEF 2011 plant images classification task. In *CLEF (Notebook Papers/Labs/Workshop)*, 2011.

[149] Hervé Goëau, Pierre Bonnet, Alexis Joly, Itheri Yahiaoui, Daniel Barthelemy, Nozha Boujemaa, and Jean-François Molino. The ImageCLEF 2012 plant identification task. In *CLEF (Online Working Notes/Labs/Workshop)*, 2012.

[150] Hervé Goëau, Pierre Bonnet, Alexis Joly, Vera Bakic, Daniel Barthelemy, Nozha Boujemaa, and Jean-François Molino. The ImageCLEF 2013 plant identification task. In *CLEF (Working Notes)*, 2013.

[151] Hervé Goëau, Alexis Joly, Pierre Bonnet, Souheil Selmi, Jean-François Molino, Daniel Barthelemy, and Nozha Boujemaa. LifeCLEF plant identification task 2014. In *CLEF (Working Notes)*, 2014.

[152] Hervé Goëau, Pierre Bonnet, and Alexis Joly. LifeCLEF plant identification task 2015. In *CLEF (Working Notes)*, 2015.

[153] Hervé Goëau, Pierre Bonnet, and Alexis Joly. Plant identification in an open-world (lifeclef 2016). In *CLEF working notes 2016*, 2016.

[154] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2015.

[155] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *Computing Research Repository (CoRR)*, 2014. arXiv: 1409.1556.

[156] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional architecture for fast feature embedding. In *Proceedings of the 22nd ACM International Conference on Multimedia*, pages 675–678, 2014.

[157] Mostafa Mehdipour-Ghazi, Berrin Yanikoglu, and Erchan Aptoula. Open-set plant identification using an ensemble of deep convolutional neural networks. In *Working Notes of CLEF 2016 - Conference and Labs of the Evaluation forum, Évora, Portugal, 5-8 September, 2016.*, pages 518–524, 2016.

[158] Merijn Van Erp and Lambert Schomaker. Variants of the Borda count method for combining ranked classifier hypotheses. In *In the seventh international workshop on frontiers in handwriting recognition*, 2000.

[159] Herve Goeau, Pierre Bonnet, and Alexis Joly. Plant identification based on noisy web data: the amazing performance of deep learning (lifeclef 2017). In *CLEF (Working Notes)*. CEUR Workshop Proceedings, 2017.

[160] Hervé Goëau, Pierre Bonnet, and Alexis Joly. Overview of expertlifeclef 2018: how far automated identification systems are from the best experts? In *CLEF working notes 2018*, 2018.

[161] Alexis Joly, Hervé Goëau, Christophe Botella, Hervé Glotin, Pierre Bonnet, Willem-Pier Vellinga, Robert Planqué, and Henning Müller. Overview of lifeclef 2018: a large-scale evaluation of species identification and recommendation algorithms in the era of ai. In *Proceedings of CLEF 2018*, 2018.

[162] Berrin Yanikoglu, Erchan Aptoula, and Caglar Tirkaz. Sabanci-Okan system at imageclef 2011: Plant identification task. In *CLEF (Working Notes)*, 2011.

[163] Berrin Yanikoglu, Erchan Aptoula, and Caglar Tirkaz. Sabanci-Okan system at imageclef 2012: Combining features and classifiers for plant identification. In *CLEF (Working Notes)*, 2012.

[164] Mostafa Mehdipour-Ghazi, Berrin Yanikoglu, and Erchan Aptoula. Plant identification using deep neural networks via optimization of transfer learning parameters. *Neurocomputing*, 235:228–235, 2017.

[165] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. *arXiv preprint arXiv:1709.01507*, 2017.

[166] Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[167] Jonathan Harel, Christof Koch, and Pietro Perona. Graph-based visual saliency. In *Advances in neural information processing systems*, pages 545–552, 2007.

[168] Cemre Zor, Berrin Yanikoglu, Erinc Merdivan, Terry Windeatt, Josef Kittler, and Ethem Alpaydin. BeamECOC: A local search for the optimization of the ECOC matrix. In *23rd International Conference on Pattern Recognition, ICPR 2016, Cancún, Mexico, December 4-8, 2016*, pages 198–203, 2016.