

# Cost-Sensitive Learning of Deep Feature Representations from Imbalanced Data

S. H. Khan, M. Bennamoun, F. Sohel and R. Togneri

**Abstract**—Class imbalance is a common problem in the case of real-world object detection and classification tasks. Data of some classes is abundant making them an over-represented majority, and data of other classes is scarce, making them an under-represented minority. This imbalance makes it challenging for a classifier to appropriately learn the discriminating boundaries of the majority and minority classes. In this work, we propose a cost sensitive deep neural network which can automatically learn robust feature representations for both the majority and minority classes. During training, our learning procedure jointly optimizes the class dependent costs and the neural network parameters. The proposed approach is applicable to both binary and multi-class problems without any modification. Moreover, as opposed to data level approaches, we do not alter the original data distribution which results in a lower computational cost during the training process. We report the results of our experiments on six major image classification datasets and show that the proposed approach significantly outperforms the baseline algorithms. Comparisons with popular data sampling techniques and cost sensitive classifiers demonstrate the superior performance of our proposed method.

**Index Terms**—Cost-sensitive learning, Convolutional Neural Networks, Data imbalance, Loss functions.

## I. INTRODUCTION

In most real-world classification problems, the collected data follows a long tail distribution i.e., data for few object classes is abundant while data for others is scarce. This behaviour is termed the ‘*class-imbalance problem*’ and it is inherently manifested in nearly all of the collected image classification databases (e.g., Fig. 1). A multi-class dataset is said to be ‘*imbalanced*’ or ‘*skewed*’ if some of its (minority) classes, in the training set, are heavily under-represented compared to other (majority) classes. This skewed distribution of class instances forces the classification algorithms to be biased towards the majority classes. As a result, the characteristics of the minority classes are not adequately learned.

The class imbalance problem is of particular interest in real world scenarios, where it is essential to correctly classify examples from an infrequent but important minority class. For instance, a particular cancerous lesion (e.g., a melanoma)

which appears rarely during dermoscopy should not be misclassified as benign (see Sec. IV). Similarly, for a continuous surveillance task, a dangerous activity which occurs occasionally should still be detected by the monitoring system. The same applies to many other application domains, e.g., object classification, where the correct classification of a minority class sample is equally important to the correct classification of a majority class sample. It is therefore required to enhance the overall accuracy of the system without unduly sacrificing the precision of any of the majority or minority classes. Most of the classification algorithms try to minimize the overall classification error during the training process. They, therefore, implicitly assign an identical misclassification cost to all types of errors assuming their equivalent importance. As a result the classifier tends to correctly classify and favour the more frequent classes.

Despite the pertinence of the class imbalance problem to practical computer vision, there have been very few research works on this topic in the recent years. Class imbalance is avoided in nearly all competitive datasets during the evaluation and training procedures (see Fig. 1). For instance, for the case of the popular image classification datasets (such as CIFAR-10/100, ImageNet, Caltech-101/256, and MIT-67), efforts have been made by the collectors to ensure that, either all of the classes have a minimum representation with sufficient data, or that the experimental protocols are reshaped to use an equal number of images for all classes during the training and testing processes [1, 2, 3]. This approach is reasonable in the case of datasets with only few classes, which have an equal probability to appear in practical scenarios (e.g., digits in MNIST). However, with the increasing number of classes in the collected object datasets, it is becoming impractical to provide equal representations for all classes in the training and testing subsets. For example, for a fine-grained coral categorization dataset, endangered coral species have a significantly lower representation compared to the more abundant ones [4].

In this work, we propose to jointly learn robust feature representations and classifier parameters, under a cost-sensitive setting. This enables us to learn not only an improved classifier that deals with the class imbalance problem, but also to extract suitably adapted intermediate feature representations from a deep Convolutional Neural Network (CNN). In this manner, we directly modify the learning procedure to incorporate class dependent costs during training. In contrast, previous works (such as [5, 6, 7, 8]) only readjust the training data distribution to learn better classifiers. Moreover, unlike the methods in e.g., [4, 9, 10], we do not use a handcrafted cost matrix

S. H. Khan and M. Bennamoun are with the School of Computer Science and Software Engineering, The University of Western Australia, 35 Stirling Highway, Crawley, WA 6009, Australia.

E-mail: {salman.khan, mohammed.bennamoun} @uwa.edu.au

R. Togneri is with the School of Electrical, Electronic and Computer Engineering, The University of Western Australia, 35 Stirling Highway, Crawley, WA 6009, Australia.

E-mail: roberto.togneri@uwa.edu.au

F. Sohel is with the School of Engineering and Information Technology, Murdoch University, 90 South St, Murdoch WA 6150, Australia.

E-mail: f.sohel@murdoch.edu.au

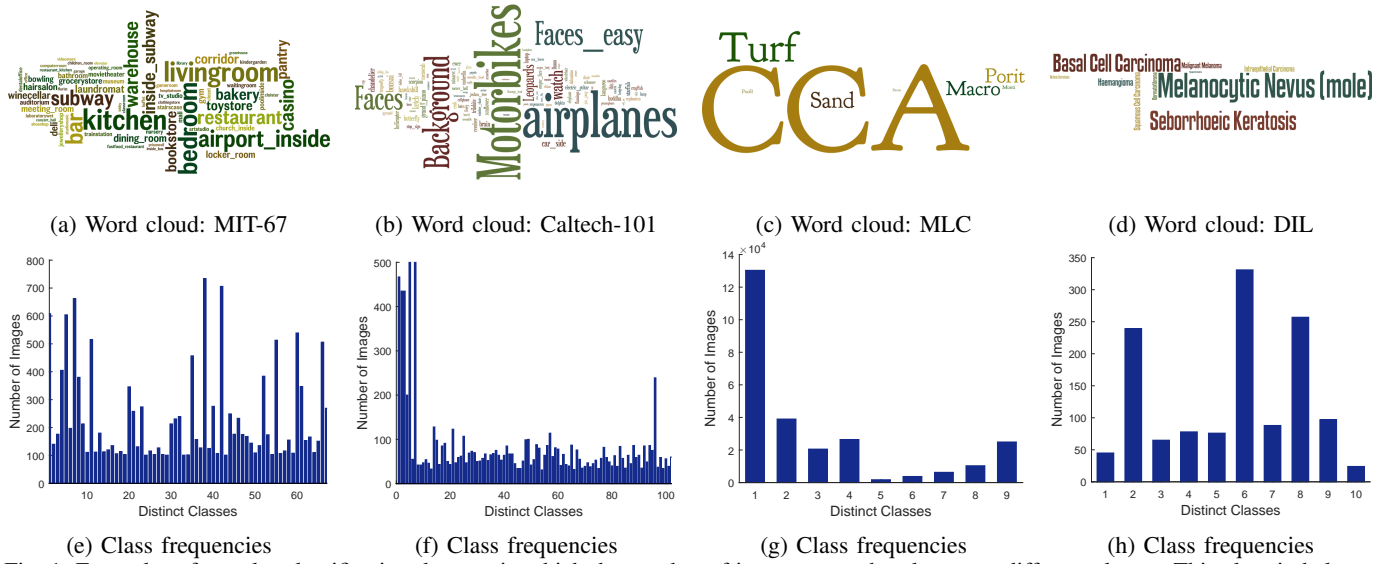


Fig. 1: Examples of popular classification datasets in which the number of images vary sharply across different classes. This class imbalance poses a challenge for classification and representation learning algorithms.

whose design is based on expert judgement and turns into a tedious task for a large number of classes. In our case, the class dependent costs are automatically set using data statistics (e.g., data distribution and separability measures) during the learning procedure. Another major difference with existing techniques is that our class specific costs are only used during the training process and once the optimal CNN parameters are learnt, predictions can be made without any modification to the trained network. From this perspective, our approach can be understood as a perturbation method, which forces the training algorithm to learn more discriminative features. Nonetheless, it is clearly different from the common perturbation mechanisms used during training e.g., data distortions [11], corrupted features [12], affine transformations [13] and activation dropout [14].

Our contribution consists of the following: **1**– We introduce cost-sensitive versions of three widely used loss functions for joint cost-sensitive learning of features and classifier parameters in the CNN (Sec. III-C). We also show that the improved loss functions have desirable properties such as classification calibration and guess-aversion. **2**– We analyse the effect of these modified loss functions on the back-propagation algorithm by deriving relations for propagated gradients (Sec. III-E). **3**– We propose an algorithm for joint alternate optimization of the network parameters and the class-sensitive costs (Sec. III-D). The proposed algorithm can automatically work for both binary and multi-class classification problems. We also show that the introduction of class-sensitive costs does not significantly affect the training and testing time of the original network (Sec. IV). **4**– The proposed approach has been extensively tested on six major classification datasets and has shown to outperform baseline procedures and state-of-the-art approaches (Sec. IV-C).

The remainder of this paper is organized as follows. We briefly discuss the related work in the next section. In Sec. III-A and III-B, we introduce our proposed approach and

analyse the modified loss functions in Sec. III-C. The learning algorithm is then described in Sec. III-D and the CNN implementation details are provided in Sec. IV-B. Experiments and results are summarized in Sec. IV and the paper concludes in Sec. V.

## II. RELATED WORK

Previous research on the class imbalance problem has concentrated mainly on two levels: the data level and the algorithmic level [15]. Below, we briefly discuss the different research efforts that tackle the class imbalance problem.

**Data level approaches** manipulate the class representations in the original dataset by either over-sampling the minority classes or under-sampling the majority classes to make the resulting data distribution balanced [15]. However, these techniques change the original distribution of the data and consequently introduce drawbacks. While under-sampling can potentially lose useful information about the majority class data, over-sampling makes the training computationally burdensome by artificially increasing the size of the training set. Furthermore, over-sampling is prone to cause over-fitting, when exact copies of the minority class are replicated randomly [5, 15].

To address the over-fitting problem, Chawla *et al.* [5] introduced a method, called SMOTE, to generate new instances by linear interpolation between closely lying minority class samples. These synthetically generated minority class instances may lie inside the convex hull of the majority class instances, a phenomenon known as *over-generalization*. Over the years, several variants of the SMOTE algorithm have been proposed to solve this problem [16]. For example, Borderline SMOTE [17] only over-samples the minority class samples which lie close to the class boundaries. Safe-level SMOTE [18] carefully generates synthetic samples in the so called *safe-regions*, where the majority and minority class regions are not overlapping. The local neighborhood SMOTE [19] considers

the neighboring majority class samples when generating synthetic minority class samples and reports a better performance compared to the former variants of SMOTE. The combination of under and over sampling procedures (e.g., [8, 20, 21]) to balance the training data have also shown to perform well. However, a drawback of these approaches is the increased computational cost, that is required for data pre-processing and for the learning of a classification model. Moreover, they are not straightforwardly applicable to multiclass problems.

**Algorithm level approaches** directly modify the learning procedure to improve the sensitivity of the classifier towards minority classes. Zhang *et al.* [7] first divided the data into smaller balanced subsets, followed by intelligent sampling and a cost-sensitive SVM learning to deal with the imbalance problem. A neuro-fuzzy modeling procedure was introduced in [22] to perform leave-one-out cross-validation on imbalanced datasets. A scaling kernel along-with the standard SVM was used in [23] to improve the generalization ability of learned classifiers for skewed datasets. Li *et al.* [24] gave more importance to the minority class samples by setting weights with Adaboost during the training of an extreme learning machine (ELM). An ensemble of soft-margin SVMs was formed via boosting to perform well on both majority and minority classes [25]. These previous works hint towards the use of distinct costs for different training examples to improve the performance of the learning algorithm. However, they do not address the class imbalance learning of CNNs, which have recently emerged as the most popular tool for supervised classification, recognition and segmentation problems in computer vision [13, 23, 26, 27, 28]. Furthermore, they are mostly limited to the binary class problems [25, 29], do not perform joint feature and classifier learning and do not explore computer vision tasks which inherently have imbalanced class distributions. In contrast, this paper presents the first attempt to incorporate automatic cost-sensitive learning in deep neural networks for imbalanced data.

### III. PROPOSED APPROACH

#### A. Problem Formulation for Cost Sensitive Classification

Let the cost  $\xi'_{p,q}$  be used to denote the misclassification cost of classifying an instance belonging to a class  $p$  into a different class  $q$ . The diagonal of  $\xi'$  (i.e.,  $\xi'_{p,p}, \forall p$ ) represents the benefit or utility for a correct prediction. Given an input instance  $\mathbf{x}$  and the cost matrix  $\xi'$ , the classifier seeks to minimize the expected risk  $\mathcal{R}(p|\mathbf{x})$ , where  $p$  is the class prediction made by the classifier. The expected risk can be expressed as:

$$\mathcal{R}(p|\mathbf{x}) = \sum_q \xi'_{p,q} P(q|\mathbf{x}),$$

where,  $P(q|\mathbf{x})$  is the posterior probability over all possible classes given an instance  $\mathbf{x}$ . According to the Bayes decision theory, an ideal classifier will give a decision in favour of the class ( $p^*$ ) with the minimum expected risk:

$$p^* = \underset{p}{\operatorname{argmin}} \mathcal{R}(p|\mathbf{x}) = \underset{p}{\operatorname{argmin}} \mathbb{E}_{X,D}[\xi'] \quad (1)$$

where,  $X$  and  $D$  define the input and output spaces respectively. Since,  $P(q|\mathbf{x})$  cannot be found trivially, we make use

of empirical distribution derived from the training data. Given a training dataset consisting of tuples comprising of data and label,  $\mathcal{D} = \{\mathbf{x}^{(i)}, \mathbf{d}^{(i)}\}_M$ , we can define the empirical risk as follows:

$$\hat{\mathcal{R}}_\ell(\mathbf{o}) = \mathbb{E}_{X,D}[\ell] = \frac{1}{M} \sum_{i=1}^M \ell(\xi', \mathbf{d}^{(i)}, \mathbf{o}^{(i)}), \quad (2)$$

where,  $M$  is the total number of images,  $\mathbf{o}^{(i)}$  is the neural network output for the  $i^{th}$  sample and  $\ell(\cdot)$  is the misclassification error (0 – 1 loss) or a surrogate loss function which is typically used during the classifier training. For the case of cost-insensitive 0–1 loss,  $\ell(\xi', \mathbf{d}^{(i)}, \mathbf{o}^{(i)}) = \mathbb{I}(\mathbf{d}^{(i)} \neq \mathbf{o}^{(i)})$  and  $\xi'$  is an  $N \times N$  matrix, where  $\xi'_{p,p} = 0$ , and  $\xi'_{p,q} = 1, \forall p \neq q$ .

**Properties of the Cost Matrix  $\xi'$ :** Lemmas III.1 and III.2 describe the main properties of the cost matrix  $\xi'$ . Their proof can be found in Appendix A (supplementary material).

**Lemma III.1.** *Offsetting the columns of the cost matrix  $\xi'$  by any constant ‘c’ does not affect the associated classification risk  $\mathcal{R}$ .*

For convenience, the utility vector (i.e., the diagonal of the cost matrix) for correct classification is usually set to zero with the help of the property from Lemma III.1. We also show next that even when the utility is not zero, it must satisfy the following condition:

**Lemma III.2.** *The cost of the true class should be less than the mean cost of all misclassifications.*

Finally, using Lemmas III.1 and III.2, we assume the following:

**Assumption.** *All costs are non-negative i.e.,  $\xi' \succeq 0$ .*

The entries of a traditional cost matrix (defined according to the properties above) usually have the form of:

$$\xi' = \begin{cases} \xi'_{p,q} = 0 & p = q \\ \xi'_{p,q} \in \mathbb{N} & p \neq q. \end{cases} \quad (3)$$

Such cost matrix can potentially increase the corresponding loss to a large value. During the CNN training, this network loss can make the training process unstable and can lead to the non-convergence of the error function. This requires the introduction of an alternative cost matrix.

#### B. Our Proposed Cost Matrix

We propose a new cost matrix  $\xi$ , which is suitable for CNN training. The cost matrix  $\xi$  is used to modify the output of the last layer of a CNN (before the softmax and the loss layer) (Fig. 2). The resulting activations are then squashed between  $[0, 1]$  before the computation of the classification loss.

For the case of a CNN, the classification decision is made in favour of the class with the maximum classification score. During the training process, the classifier weights are modified in order to reshape the classifier confidences (class probabilities) such that the desired class has the maximum score and the other classes have a considerably lower score. However, since the less frequent classes are under-represented in the training set, we introduce new ‘score-level costs’ to encourage

the correct classification of infrequent classes. Therefore the CNN outputs are modified as follows:

$$\mathbf{s}_p^{(i)} = \xi_p \circ \mathbf{o}^{(i)}$$

$$\forall j, \quad s_{p,j} = \xi_{p,j} o_{j,p} \quad | \quad s_{p,p} \geq s_{p,j}, \quad \forall j \neq p,$$

where,  $p$  is the desired class and  $\circ$  denotes the Hadamard product. Note that the score level costs perturb the classifier confidences. Such perturbation allows the classifier to give more importance to the less frequent and difficult-to-separate classes.

**Properties of the Proposed Cost Matrix  $\xi$ :** Next, we discuss few properties (lemmas III.3 – III.6) of the newly introduced cost matrix  $\xi$  and its similarities/differences with the traditionally used cost matrix  $\xi'$  (Sec. III-A):

**Lemma III.3.** *The cost matrix  $\xi$  for a cost-insensitive loss function is an all-ones matrix,  $\mathbf{1}^{p \times p}$ , rather than a  $\mathbf{1} - \mathbf{I}$  matrix, as in the case of the traditionally used cost matrix  $\xi'$ .*

*Proof:* With all costs equal to the multiplicative identity i.e.,  $\xi_{p,q} = 1$ , the CNN activations will remain unchanged. Therefore, all decisions have a uniform cost of 1 and the classifier is cost-insensitive. ■

**Lemma III.4.** *All costs in  $\xi$  are positive, i.e.,  $\xi \succ 0$ .*

*Proof:* We adopt a proof by contradiction. Let us suppose that  $\xi_{p,q} = 0$ . During training in this case, the corresponding score for class  $q$  ( $s_{p,q}$ ) will always be zero for all samples belonging to class  $p$ . As a result, the output activation ( $y_q$ ) and the back-propagated error will be independent of the weight parameters of the network, which proves the Lemma. ■

**Lemma III.5.** *The cost matrix  $\xi$  is defined such that all of its elements in are within the range  $(0, 1]$ , i.e.,  $\xi_{p,q} \in (0, 1]$ .*

*Proof:* Based on Lemmas III.3 and III.4, it is trivial that the costs are with-in the range  $(0, 1]$ . ■

**Lemma III.6.** *Offsetting the columns of the cost matrix  $\xi$  can lead to an equally probable guess point.*

*Proof:* Let us consider the case of a cost-insensitive loss function. In this case,  $\xi = \mathbf{1}$  (from Lemma III.3). Offsetting all of its columns by a constant  $c = 1$  will lead to  $\xi = \mathbf{0}$ . For  $\xi = \mathbf{0}$ , the CNN outputs will be zero for any  $\mathbf{o}^{(i)} \in \mathbb{R}^N$ . Therefore, the classifier will make a random guess for classification. ■

The cost matrix  $\xi$  configured according to the properties described above (Lemma III.3 – III.6) neither excessively increases the CNN outputs activations, nor does it reduce them to zero output values. This enables a smooth training process allowing the model parameters to be correctly updated. In the following section, we analyse the implications of the newly introduced cost matrix  $\xi$  on the loss layer (Fig. 2).

### C. Cost-Sensitive Surrogate Losses

Our approach addresses the class imbalance problem during the training of CNNs. For this purpose, we introduce a cost sensitive error function which can be expressed as the mean

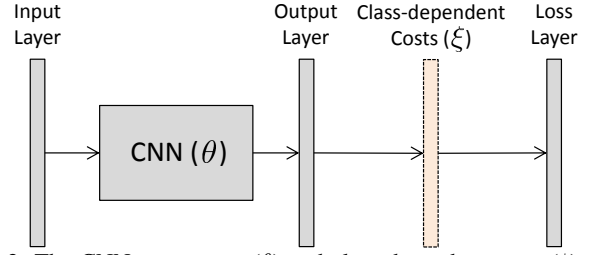


Fig. 2: The CNN parameters ( $\theta$ ) and class dependent costs ( $\xi$ ) used during the training process of our deep network. Details about the CNN architecture and the loss layer are in Sec. IV-B and III-C, respectively

loss over the training set:

$$E(\theta, \xi) = \frac{1}{M} \sum_{i=1}^M \ell(\mathbf{d}^{(i)}, \mathbf{y}_{\theta, \xi}^{(i)}), \quad (4)$$

where, the predicted output ( $\mathbf{y}$ ) of the penultimate layer (before the loss layer) is parameterized by  $\theta$  (network weights and biases) and  $\xi$  (class sensitive costs),  $M$  is the total number of training examples,  $\mathbf{d} \in \{0, 1\}^{1 \times N}$  is the desired output (s.t.  $\sum_n d_n := 1$ ) and  $N$  denotes the total number of neurons in the output layer. For conciseness, we will not explicitly mention the dependence of  $\mathbf{y}$  on the parameters  $(\theta, \xi)$  and only consider a single data instance in the discussion below. Note that the error is larger when the model performs poorly on the training set. The objective of the learning algorithm is to find the optimal parameters  $(\theta^*, \xi^*)$  which give the minimum possible cost  $E^*$  (Eq. (4)). Therefore, the optimization objective is given by:

$$(\theta^*, \xi^*) = \arg \min_{\theta, \xi} E(\theta, \xi). \quad (5)$$

The loss function  $\ell(\cdot)$  in Eq. (4) can be any suitable surrogate loss such as the Mean Square Error (MSE), Support Vector Machine (SVM) hinge loss or a Cross Entropy (CE) loss (also called the ‘soft-max log loss’). These popular loss functions are shown along-with other surrogate losses in Fig. 3. The cost sensitive versions of these loss functions are discussed below:

**(a) Cost Sensitive MSE loss:** This loss minimizes the squared error of the predicted output with the desired ground-truth and can be expressed as follows:

$$\ell(\mathbf{d}, \mathbf{y}) = \frac{1}{2} \sum_n (d_n - y_n)^2 \quad (6)$$

where,  $y_n$  is related to the output of the previous layer  $o_n$  via the logistic function,

$$y_n = \frac{1}{1 + \exp(-o_n \xi_{p,n})}, \quad (7)$$

where,  $\xi$  is the class sensitive penalty which depends on the desired class of a particular training sample, i.e.,  $p = \arg \max_m d_m$ . The effect of this cost on the back-propagation algorithm is discussed in Sec. III-E1.

**(b) Cost Sensitive SVM hinge loss:** This loss maximizes the margin between each pair of classes and can be expressed as

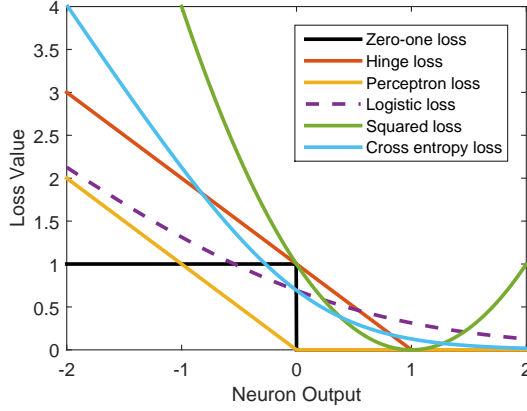


Fig. 3: This figure shows the 0-1 loss along-with several other common surrogate loss functions that are used for binary classification.

follows:

$$\ell(\mathbf{d}, \mathbf{y}) = - \sum_n \max(0, 1 - (2d_n - 1)y_n), \quad (8)$$

where,  $y_n$  can be represented in terms of the previous layer output  $o_n$  and the cost  $\xi$ , as follows:

$$y_n = o_n \xi_{p,n}. \quad (9)$$

The effect of the introduced cost on the gradient computation is discussed in Sec. III-E2.

**(c) Cost Sensitive CE loss:** This loss maximizes the closeness of the prediction to the desired output and is given by:

$$\ell(\mathbf{d}, \mathbf{y}) = - \sum_n (d_n \log y_n), \quad (10)$$

where  $y_n$  incorporates the class dependent cost ( $\xi$ ) and is related to the output  $o_n$  via the soft-max function,

$$y_n = \frac{\xi_{p,n} \exp(o_n)}{\sum_k \xi_{p,k} \exp(o_k)}. \quad (11)$$

The effect of the modified CE loss on the back-propagation algorithm is discussed in Sec. III-E3.

**Classification Feasibility of Cost-Sensitive Losses::** Next, we show (Lemmas III.7–III.9) that the cost-sensitive loss functions remain suitable for classification since they satisfy the following properties:

- 1) Classification Calibration [30]
- 2) Guess Aversion [4]

Note that a classification calibrated (c-calibrated) loss is useful because the minimization of the empirical risk leads to classifiers which have risks that are closer to the Bayes-risk. Similarly, guess aversion implies that the loss function favours ‘correct classification’ instead of ‘arbitrary guesses’. Since, CE loss usually performs best among the three loss functions we discussed above, Lemmas III.7–III.9 show that the cost sensitive CE loss is guess averse and classification calibrated.

**Lemma III.7.** For a real valued  $\xi$  ( $\xi \in \mathbb{R}^{C \times C} \in (0, 1]$ ), given  $\mathbf{d}^{(i)}$  and the CNN output  $\mathbf{o}^{(i)}$ , the modified cost sensitive CE loss will be guess-averse iff,

$$L(\xi, \mathbf{d}, \mathbf{o}) < L(\xi, \mathbf{d}, \mathbf{g}),$$

where,  $\mathbf{g}$  is the set of all guess points.

*Proof:* For real valued CNN activations, the guess point maps to an all zero output:

$$L(\xi, \mathbf{d}, \mathbf{o}) < L(\xi, \mathbf{d}, \mathbf{0}),$$

$$-\log \left( \frac{\xi_{p,n} \exp(o_n)}{\sum_k \xi_{p,k} \exp(o_k)} \right) < -\log \left( \frac{\xi_{p,n}}{\sum_k \xi_{p,k}} \right),$$

which can be satisfied if,

$$\frac{\xi_{p,n} \exp(o_n)}{\sum_k \xi_{p,k} \exp(o_k)} > \frac{\xi_{p,n}}{\sum_k \xi_{p,k}}.$$

where,  $n$  is the true class. Since,  $\xi_{p,n} \in (0, 1]$  and thus it is  $> 0$ . Also, if  $n$  is the true class then  $o_n > o_k, \forall k \neq n$ . Therefore, the above relation holds true. ■

**Lemma III.8.** The cost matrix has diagonal entries greater than zero, i.e.,  $\text{diag}(\xi) > 0$ .

*Proof:* According to Lemma III.1, if the CE loss is guess averse, it must satisfy ,

$$L(\xi, \mathbf{d}, \mathbf{o}) < L(\xi, \mathbf{d}, \mathbf{0}).$$

We prove the Lemma by contradiction. Let us suppose that  $\xi_{p,n} = 0$ , then the above relation does not hold true, since:

$$\frac{\xi_{p,n} \exp(o_n)}{\sum_k \xi_{p,k} \exp(o_k)} = \frac{\xi_{p,n}}{\sum_k \xi_{p,k}} = 0.$$

and hence,  $\text{diag}(\xi) > 0$ . ■

**Lemma III.9.** The cost sensitive CE loss function

$$\ell(\xi, \mathbf{d}, \mathbf{o}) = - \sum_n d_n \log \left( \frac{\xi_{p,n} \exp(o_n)}{\sum_k \xi_{p,k} \exp(o_k)} \right),$$

is C-Calibrated.

*Proof:* Given an input sample  $x$  which belongs to class  $p^*$  (i.e.,  $d_{p^*} = 1$ ), then the CE loss can be expressed as:

$$\ell(\xi, \mathbf{d}, \mathbf{o}) = - \log \left( \frac{\xi_{p^*,p^*} \exp(o_{p^*})}{\sum_k \xi_{p,k} \exp(o_k)} \right)$$

The classification risk can be expressed in terms of the expected value as follows:

$$\begin{aligned} \mathcal{R}_\ell[\mathbf{o}] &= \mathbb{E}_{X,D}[\ell(\xi, \mathbf{d}, \mathbf{o})] \\ &= \sum_{p=1}^N P(p|\mathbf{x}) \ell(\xi, \mathbf{d}, \mathbf{o}) = \sum_{p=1}^N P(p|\mathbf{x}) \ell(\xi, p, \mathbf{o}) \\ &= - \sum_{p=1}^N P(p|\mathbf{x}) \log \left( \frac{\xi_{p,p} \exp(o_p)}{\sum_k \xi_{p,k} \exp(o_k)} \right) \end{aligned}$$

Next, we compute the derivative and set it to zero to find the ideal set of CNN outputs ‘o’,

$$\begin{aligned} \frac{\partial \mathcal{R}_\ell[\mathbf{o}]}{\partial o_t} &= \frac{\partial \mathcal{R}_\ell[o_p]}{\partial o_t} \Big|_{p \neq t} + \frac{\partial \mathcal{R}_\ell[o_p]}{\partial o_t} \Big|_{p=t} = 0 \\ \frac{\partial \mathcal{R}_\ell[o_p]}{\partial o_t} \Big|_{p \neq t} &= \frac{\partial}{\partial o_t} \left( - \sum_{p=1}^N P(p|\mathbf{x}) \log(\xi_{p,p} \exp(o_p)) \right. \\ &\quad \left. + \sum_{p=1}^N P(p|\mathbf{x}) \log\left(\sum_k \xi_{p,k} \exp(o_k)\right) \right) \\ &= -P(t|\mathbf{x}) + P(t|\mathbf{x}) \frac{\xi_{t,t} \exp(o_t)}{\sum_{k=1}^N \xi_{t,k} \exp(o_k)} \end{aligned}$$

Similarly,

$$\frac{\partial \mathcal{R}_\ell[o_p]}{\partial o_t} \Big|_{p \neq t} = \sum_{p \neq t} P(p|\mathbf{x}) \frac{\xi_{p,t} \exp(o_t)}{\sum_{k=1}^N \xi_{p,k} \exp(o_k)}$$

By adding the above two derived expression and setting them to zero, we have :

$$\begin{aligned} P(t|\mathbf{x}) &= \exp(o_t) \sum_{p=1}^N \frac{P(p|\mathbf{x}) \xi_{p,t}}{\sum_{k=1}^N \xi_{p,k} \exp(o_k)} \\ o_t &= \log(P(t|\mathbf{x})) - \log\left(\sum_{p=1}^N P(p|\mathbf{x}) \xi_{p,t}\right) \\ &\quad + \log\left(\sum_{p=1}^N \sum_{k=1}^N \xi_{p,k} \exp(o_k)\right) \end{aligned}$$

Which shows that there exists an inverse relationship between the optimal CNN output and the Bayes cost of the  $t^{th}$  class, and hence, the cost-sensitive CE loss is classification calibrated. ■

Under the properties of Lemmas III.7–III.9, the modified loss functions are therefore suitable for classification. Having established the class dependent costs (Sec. III-B) and their impact on the loss layer (Sec. III-C), we next describe the training algorithm to automatically learn all the parameters of our model ( $\theta$  and  $\xi$ ).

#### D. Optimal Parameters Learning

When using any of the previously mentioned loss functions (Eqs. (6-10)), our goal is to jointly learn the hypothesis parameters  $\theta$  and the class dependent loss function parameters  $\xi$ . For the joint optimization, we alternatively solve for both types of parameters by keeping one fixed and minimizing the cost with respect to the other (Algorithm 1). Specifically, for the optimization of  $\theta$ , we use the stochastic gradient descent (SGD) with the back-propagation of error (Eq. (4)). Next, to optimize  $\xi$ , we again use the gradient descent algorithm to calculate the direction of the step to update the parameters. The cost function is also dependent on the class-to-class separability, the current classification errors made by the network with current estimate of parameters and the overall

---

#### Algorithm 1 Iterative optimization for parameters ( $\theta, \xi$ )

---

**Input:** Training set ( $\mathbf{x}, \mathbf{d}$ ), Validation set ( $\mathbf{x}_V, \mathbf{d}_V$ ), Max. epochs ( $E$ ), Learning rate for  $\theta$  ( $\gamma_\theta$ ), Learning rate for  $\xi$  ( $\gamma_\xi$ )

**Output:** Learned parameters ( $\theta^*, \xi^*$ )

```

1: Net ← construct_CNN()
2:  $\theta \leftarrow \text{initialize\_Net}(\text{Net})$   $\triangleright$  Random initialization
3:  $\xi \leftarrow \mathbf{1}$ , val-err  $\leftarrow 1$ 
4: for  $e \in [1, E]$  do  $\triangleright$  Number of epochs
5:    $\text{grad}_\xi \leftarrow \text{compute\_gard}(\mathbf{x}, \mathbf{d}, F(\xi))$   $\triangleright$  Eq. (15)
6:    $\xi^* \leftarrow \text{update\_CostParams}(\xi, \gamma_\xi, \text{grad}_\xi)$ 
7:    $\xi \leftarrow \xi^*$ 
8:   for  $b \in [1, B]$  do  $\triangleright$  Number of batches
9:      $\text{out}_b \leftarrow \text{forward-pass}(\mathbf{x}_b, \mathbf{d}_b, \text{Net}, \theta)$ 
10:     $\text{grad}_b \leftarrow \text{backward-pass}(\text{out}_b, \mathbf{x}_b, \mathbf{d}_b, \text{Net}, \theta, \xi)$ 
11:     $\theta^* \leftarrow \text{update\_NetParams}(\text{Net}, \theta, \gamma_\theta, \text{grad}_b)$ 
12:     $\theta \leftarrow \theta^*$ 
13:   end for
14:   val-err*  $\leftarrow \text{forward-pass}(\mathbf{x}_V, \mathbf{d}_V, \text{Net}, \theta)$ 
15:   if val-err*  $>$  val-err then
16:      $\gamma_\xi \leftarrow \gamma_\xi * 0.01$   $\triangleright$  Decrease step size
17:     val-err  $\leftarrow \text{val-err}^*$ 
18:   end if
19: end for
20: return ( $\theta^*, \xi^*$ )
```

---

classification error. The class-to-class (c2c) separability is measured by estimating the spread of the with-in class samples (intra-class) compared to the between-class (interclass) ones. In other words, it measures the relationship between the with-in class sample distances and the size of the separating boundary between the different classes.

To calculate the c2c separability, we first compute a suitable distance measure between each point in a class  $c_p$  and its nearest neighbour belonging to  $c_p$  and the nearest neighbour in class  $c_q$ . Note that these distances are calculated in the feature space where each point is a 4096 dimensional feature vector ( $f_i, i \in [1, N]$ ) obtained from the penultimate CNN layer (just before the output layer). Next, we find the average of intra-class distances to interclass distance for each point in a class and compute the ratio of the averages to find the c2c separability index. Formally, the class separability between two classes,  $p$  and  $q$  is defined as:

$$S(p, q) = \frac{1}{N} \sum_i \frac{\text{dist}_{\text{intraNN}}(f_i)}{\text{dist}_{\text{interNN}}(f_i)}$$

To avoid over-fitting and to keep this step computationally feasible, we measure the c2c separability on a small validation set. Also, the c2c separability was found to correlate well with the confusion matrix at each epoch. Therefore the measure was calculated after every 10 epochs to minimize the computational overhead. Note that by simply setting the parameters ( $\xi$ ) based on the percentages of the classes in the data distribution results in a poor performance (Sec. IV-C). This suggests that the optimal parameter values for class dependent costs ( $\xi^*$ ) should not be the same as the frequency of the classes in the training

data distribution. The following cost function is used for the gradient computation to update  $\xi$ :

$$F(\xi) = \|T - \xi\|_2^2 + E_{val}(\theta, \xi), \quad (12)$$

The matrix  $T$  is defined as follows:

$$T = H \circ \exp\left(-\frac{(S - \mu_1)^2}{2\sigma_1^2}\right) \circ \exp\left(-\frac{(M - \mu_2)^2}{2\sigma_2^2}\right),$$

where,  $\mu, \sigma$  denote the parameters which are set using cross validation,  $M$  denotes the current classification errors as a confusion matrix,  $S$  denotes the class c2c separability matrix and  $H$  is a matrix defined using the histogram vector  $\mathbf{h}$  which encodes the distribution of classes in the training set. The matrix  $H$  and vector  $\mathbf{h}$  are linked as follows:

$$H(p, q) = \begin{cases} \max(h_p, h_q) & : p \neq q, (p, q) \in \mathbf{c}, \\ h_p & : p = q, p \in \mathbf{c} \end{cases} \quad (13)$$

where,  $\mathbf{c}$  is the set of all classes in a given dataset. The resulting minimization objective to find the optimal  $\xi^*$  can be expressed as:

$$\xi^* = \arg \min_{\xi} F(\xi). \quad (14)$$

In order to optimize the cost function in Eq. (14), we use the gradient descent algorithm which computes the direction of the update step, as follows:

$$\begin{aligned} \nabla F(\xi) &= \nabla(\mathbf{v}_a - \mathbf{v}_b)(\mathbf{v}_a - \mathbf{v}_b)^T \\ &= (\mathbf{v}_a - \mathbf{v}_b)J_{\mathbf{v}_b}^T = -(\mathbf{v}_a - \mathbf{v}_b)\mathbf{1}^T. \end{aligned} \quad (15)$$

where,  $\mathbf{v}_a = \text{vec}(T)$ ,  $\mathbf{v}_b = \text{vec}(\xi)$  and  $J$  denotes the Jacobin matrix. Note that in order to incorporate the dependence of  $F(\xi)$  on the validation error  $E_{val}$ , we take the update step only if it results in a decrease in  $E_{val}$  (see Algorithm 1).

Since, our approach involves the use of modified loss functions during the CNN parameter learning process, we will discuss their effect on the back-propagation algorithm in the next section.

### E. Effect on Error Back-propagation

In this section, we discuss the impact of the modified loss functions on the gradient computation of the back-propagation algorithm.

1) *Cost Sensitive MSE*: During the supervised training, the MSE loss minimizes the mean squared error between the predicted weighted outputs of the model  $y$ , and the ground-truth labels  $d$ , across the entire training set (Eq. (6)). The modification of the loss function changes the gradient computed during the back-propagation algorithm. Therefore, for the output layer, the mathematical expression of the gradient at each neuron is given by:

$$\frac{\partial \ell(\mathbf{d}, \mathbf{y})}{\partial o_n} = -(d_n - y_n) \frac{\partial y_n}{\partial o_n}$$

The  $y_n$  for the cost sensitive MSE loss can be defined as:

$$y_n = \frac{1}{1 + \exp(-o_n \xi_{p,n})}$$

The partial derivative can be calculated as follows:

$$\begin{aligned} \frac{\partial y_n}{\partial o_n} &= \frac{\xi_{p,n} \exp(-o_n \xi_{p,n})}{(1 + \exp(-o_n \xi_{p,n}))^2} \\ &= \frac{\xi_{p,n}}{\exp(o_n \xi_{p,n}) + \exp(-o_n \xi_{p,n}) + 2} \\ &= \frac{\xi_{p,n}}{(1 + \exp(o_n \xi_{p,n}))(1 + \exp(-o_n \xi_{p,n}))} \\ \frac{\partial y_n}{\partial o_n} &= \xi_{p,n} y_n (1 - y_n) \end{aligned}$$

The derivative of the loss function is therefore given by:

$$\frac{\partial \ell(d, y)}{\partial o_n} = -\xi_{p,n} (d_n - y_n) y_n (1 - y_n). \quad (16)$$

2) *Cost Sensitive SVM Hinge Loss*: For the SVM hinge loss function given in Eq. (8), the directional derivative can be computed at each neuron as follows:

$$\frac{\partial \ell(\mathbf{d}, \mathbf{y})}{\partial o_n} = -(2d_n - 1) \frac{\partial y_n}{\partial o_n} \mathbb{I}\{1 > y_n(2d_n - 1)\}.$$

The partial derivative of the output of the softmax w.r.t the output of the penultimate layer is given by:

$$\frac{\partial y_n}{\partial o_n} = \xi_{p,n}$$

By combining the above two expressions, the derivative of the loss function can be represented as:

$$\frac{\partial \ell(\mathbf{d}, \mathbf{y})}{\partial o_n} = -(2d_n - 1) \xi_{p,n} \mathbb{I}\{1 > y_n(2d_n - 1)\}. \quad (17)$$

where,  $\mathbb{I}(\cdot)$  denotes an indicator function.

3) *Cost Sensitive CE loss*: The cost sensitive softmax log loss function is defined in Eq. (10). Next, we show that the introduction of a cost in the CE loss does not change the gradient formulas and the cost is rather incorporated implicitly in the softmax output  $y_m$ . The effect of costs on the CE loss surface is illustrated in Fig. 4.

**Proposition 1.** *The introduction of a class imbalance cost  $\xi(\cdot)$  in the soft max loss ( $\ell(\cdot)$  in Eq. 10), does not affect the computation of the gradient during the back-propagation process.*

*Proof:* We start with the calculation of the partial derivative of the softmax neuron with respect to its input:

$$\frac{\partial y_n}{\partial o_m} = \frac{\partial}{\partial o_m} \left( \frac{\xi_{p,n} \exp(o_n)}{\sum_k \xi_{p,k} \exp(o_k)} \right) \quad (18)$$

Now, two cases can arise here, either  $m = n$  or  $m \neq n$ . We first solve for the case when  $n = m$ :

$$LHS = \xi_{p,m} \left( \frac{\exp(o_m) \sum_k \xi_{p,k} \exp(o_k) - \xi_{p,m} \exp(2o_m)}{\left( \sum_k \xi_{p,k} \exp(o_k) \right)^2} \right).$$

After simplification we get:

$$\frac{\partial y_n}{\partial o_m} = y_m (1 - y_m), \quad \text{s.t. : } m = n$$



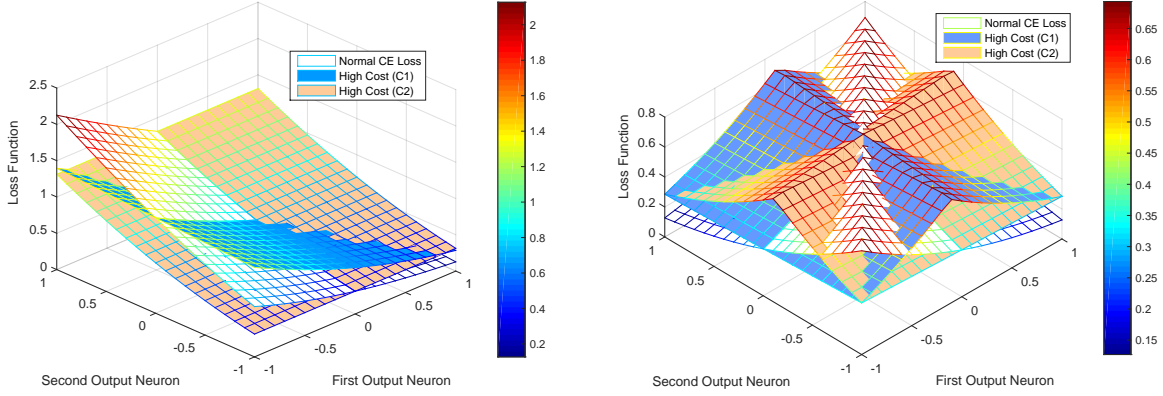


Fig. 4: The CE loss function for the case of binary classification. *Left*: loss surface for a single class for different costs (high cost for first class (C1), no cost, high cost for second class (C2)). *Right*: the minimum loss values for all possible values of class scores illustrate the obvious classification boundaries. The score-level costs reshape the loss surface and the classification boundaries are effectively shifted in favour of the classes with relatively lower cost.

Next, we solve for the case when  $n \neq m$ :

$$LHS = -\frac{\xi_{p,n}\xi_{p,n}\exp(o_m)\exp(o_n)}{\left(\sum_k \xi_{p,k}\exp(o_k)\right)^2}.$$

After simplification we get:

$$\frac{\partial y_n}{\partial o_m} = -y_m y_n, \quad s.t. : m \neq n$$

The loss function can be differentiated as follows:

$$\begin{aligned} \frac{\partial \ell(\mathbf{y}, \mathbf{d})}{\partial o_m} &= -\sum_n d_n \frac{1}{y_n} \frac{\partial y_n}{\partial o_m}, \\ &= -d_m(1 - y_m) + \sum_{n \neq m} d_n y_m = -d_m + \sum_n d_n y_m. \end{aligned}$$

Since,  $d$  is defined as a probability distribution over all output classes ( $\sum_n d_n = 1$ ), therefore:

$$\frac{\partial \ell(\mathbf{y}, \mathbf{d})}{\partial o_m} = -d_m + y_m.$$

This result is the same as in the case when CE does not contain any cost sensitive parameters. Therefore the costs affect the softmax output  $y_m$  but the gradient formulas remain unchanged. ■

In our experiments (Sec. IV), we will only report performances with the cost sensitive CE loss function. This is because, it has been shown that the CE loss outperforms the other two loss functions in most cases [31]. Moreover, it avoids the learning slowing down problem of the MSE loss [32].

#### IV. EXPERIMENTS AND RESULTS

The class imbalance problem is present in nearly all real-world object and image datasets. This is not because of any flawed data collection, but it is simply due to the natural frequency patterns of different object classes in real life. For example, a *bed* appears in nearly every bedroom scene, but a *baby cot* appears much less frequently. Consequently, from the perspective of class imbalance, the currently available image classification datasets can be divided into three categories:

- 1) Datasets with a significant class imbalance both in the training and the testing split (e.g., DIL, MLC),
- 2) Datasets with unbalanced class distributions but with experimental protocols that are designed in a way that an equal number of images from all classes are used during the training process (e.g., MIT-67, Caltech-101). The testing images can be equal or unequal for different classes.
- 3) Datasets with an equal representation of each class in the training and testing splits (e.g., MNIST, CIFAR-100).

We perform extensive experiments on six challenging image classification datasets (two from each category) (see Sec. IV-A). For the case of imbalanced datasets ( $1^{st}$  category), we report results on the standard splits for two experiments. For the two datasets from the  $2^{nd}$  category, we report our performances on the standard splits, deliberately deformed splits and the original data distributions. For the two datasets from the  $3^{rd}$  category, we report results on the standard splits and on deliberately imbalanced splits. Since, our training procedure requires a small validation set (Algorithm 1), we use  $\sim 5\%$  of the training data in each experiment as a held-out validation set.

##### A. Datasets and Experimental Settings

1) **Imbalanced Datasets: Melanoma Detection : Edinburgh Dermofit Image Library (DIL)** consists of 1300 high quality skin lesion images based on diagnosis from dermatologists and dermatopathologists. There are 10 types of lesions identified in this dataset including melanomas, seborrheic keratosis and basal cell carcinomas. The number of images in each category varies between 24 and 331 (mean 130, median 83). Similar to [33], we report results with 3-fold cross validation.

**Coral Classification : Moorea Labelled Corals (MLC)** contains 2055 images from three coral reef habitats during 2008-10. Each image is annotated with roughly 200 points belonging to the 9 classes (4 non-corals, 5 corals). Therefore in total, there are nearly 400,000 labelled points. The class



representation varies approximately from 2622 to 196910 (mean 44387, median 30817). We perform two of the major standard experiments on this dataset similar to [4]. The first experiment involves training and testing on data from year 2008. In the second experiment, training is carried out on data from year 2008 and testing on data from year 2009.

2) *Imbalanced Datasets-Balanced Protocols: Object Classification: Caltech-101* contains a total of 9,144 images, divided into 102 categories (101 objects + background). The number of images for each category varies between 31 and 800 images (mean: 90, median 59). The dataset is originally imbalanced but the standard protocol which is balanced uses 30 or 15 images for each category during training, and testing is performed on the remaining images (max. 50). We perform experiments using the standard 60%/40% and 30%/70% train/test splits.

**Scene Classification: MIT-67** consists of 15,620 images belonging to 67 classes. The number of images varies between 101 and 738 (mean: 233, median: 157). The standard protocol uses a subset of 6700 images (100 per class) for training and evaluation to make the distribution uniform. We will, however, evaluate our approach both on the standard split (80 images for training, 20 for testing) and the complete dataset with imbalanced train/test splits of 60%/40% and 30%/70%.

3) *Balanced Datasets-Balanced Protocols: Handwritten Digit Classification: MNIST* consists of 70,000 images of digits (0-9). Out of the total, 60,000 images are used for training ( $\sim 600$ /class) and the remaining 10,000 for testing ( $\sim 100$ /class). We evaluate our approach on the standard split as well as the deliberately imbalanced splits. To imbalance the training distribution, we reduce the representation of even and odd digit classes to only 25% and 10% of images, respectively.

**Image Classification: CIFAR-100** contains 60,000 images belonging to 100 classes (600 images/class). The standard train/test split for each class is 500/100 images. We evaluate our approach on the standard split as well as on artificially imbalanced splits. To imbalance the training distribution, we reduce the representation of even-numbered and odd-numbered classes to only 25% and 10% of images, respectively.

### B. Convolutional Neural Network

We use a deep CNN to learn robust feature representations for the task of image classification. The network architecture consists of a total of 18 weight layers (see Fig. 5 for details). Our architecture is similar to the state-of-the-art CNN (configuration D) proposed in [34], except that our architecture has two extra fully connected layers before the output layer and the proposed loss layer is cost sensitive. Since there are a huge number of parameters ( $\sim 139$  million) in the network, its not possible to learn all of them from scratch using a relatively smaller number of images. We, therefore, initialize the first 16 layers of our model with the pre-trained model of [34] and set random weights for the last two fully connected layers. We then train the full network with a relatively higher learning rate to allow a change in the network parameters. Note that the cost sensitive (CoSen) CNN is trained with the modified cost functions introduced in Eqs. (6-11). The CNN trained

Methods (using stand. split)	Performances	
	Exp. 1 (5-classes)	Exp. 2 (10-classes)
Hierarchical-KNN [33]	74.3 $\pm$ 2.5%	68.8 $\pm$ 2.0%
Hierarchical-Bayes [35]	69.6 $\pm$ 0.4%	63.1 $\pm$ 0.6%
Flat-KNN [33]	69.8 $\pm$ 1.6%	64.0 $\pm$ 1.3%
Baseline CNN	75.2 $\pm$ 2.7%	69.5 $\pm$ 2.3%
CoSen CNN	<b>80.2 <math>\pm</math> 2.5%</b>	<b>72.6 <math>\pm</math> 1.6%</b>

TABLE I: Evaluation on DIL Database.

Methods (using stand. split)	Performances	
	Exp. 1 (2008)	Exp. 2 (2008-2009)
MTM-CCS (LAB) [4]	74.3%	67.3%
MTM-CCS (RGB) [4]	72.5%	66.0%
Baseline CNN	72.9%	66.1%
CoSen CNN	<b>75.2%</b>	<b>68.6%</b>

TABLE II: Evaluation on MLC Database.

without cost sensitive loss layer will be used as the baseline CNN in our experiments.

### C. Results and Comparisons

For the two imbalanced datasets with imbalanced protocols, we summarize our experimental results and comparisons in Tables I, II. For each of the two datasets, we perform two standard experiments following the works of Beijbom *et al.* [4] and Ballerini *et al.* [35]. In the first experiment on the DIL dataset, we perform 3-fold cross validation on the 5 classes (namely Actinic Keratosis, Basal Cell Carcinoma, Melanocytic Nevus, Squamous Cell Carcinoma and Seborrheic Keratosis) comprising of a total of 960 images. In the second experiment, we perform 3-fold cross validation on all of the 10 classes in the DIL dataset. We achieved a performance boost of  $\sim 5.0\%$  and  $\sim 3.1\%$  over the baseline CNN in the first and second experiments respectively (Table I).

For the MLC dataset, in the first experiment we train on two-thirds of the data from 2008 and test on the remaining one third. In the second experiment, data from year 2008 is used for training and tests are performed on data from year 2009. Note that in contrast to the ‘multiple texton maps’ (MTM) [4] approach which extracts features from multiple scales, we only extract features from the  $224 \times 224$  dimensional patches. While we can achieve a larger gain by using multiple scales with our approach, we kept the setting similar to the one used with the other datasets for consistency. For similar reasons, we used the RGB color space instead of LAB, which was shown to perform better on the MLC dataset [4]. Compared to the baseline CNN, we achieved a gain of 2.3% and 2.5% on the first and second experiments respectively. Although the gains in the overall accuracy may seem modest, it should be noted that the boost in the average class accuracy is more pronounced. For example, the confusion matrices for DIL and MLC datasets in Fig. 6 (corresponding to Exp. 1 and Exp. 2 respectively), show an improvement of 9.5% and 11.8% in the average class accuracy. The confusion matrices in Figs. 6a, 6b, 6c and 6d also show a very significant boost in performance for the least frequent classes e.g., Turf, Macro, Monti, AK and SCC.

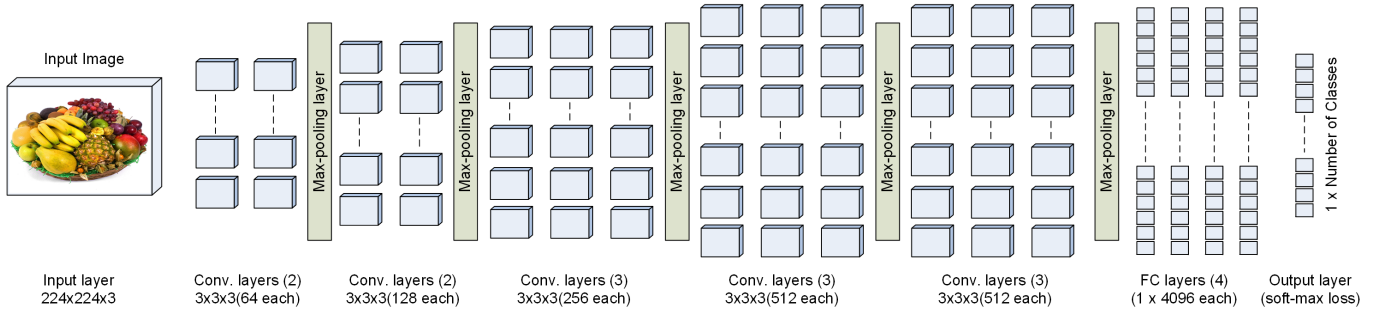


Fig. 5: The CNN architecture used in this work consists of 18 weight layers.

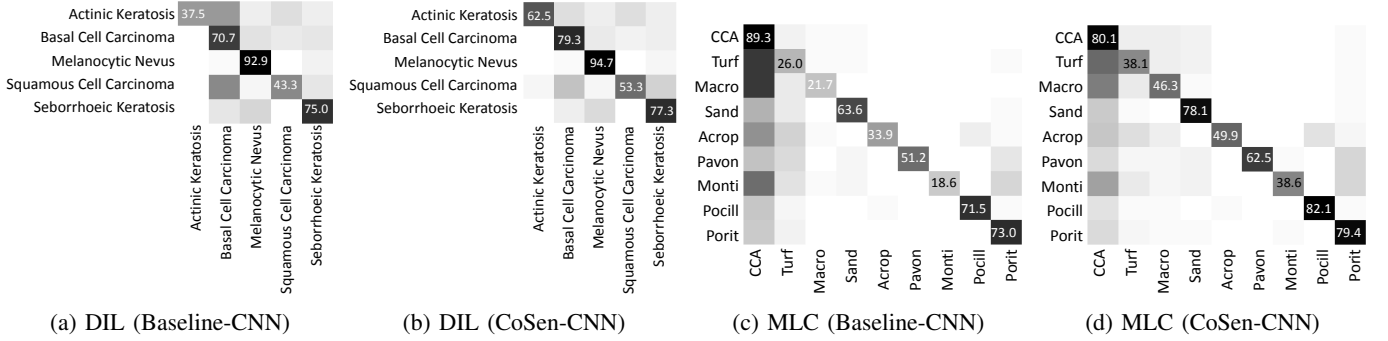


Fig. 6: Confusion Matrices for the Baseline and CoSen CNNs on the DIL and MLC datasets. Results are reported for Experiments 1 and 2 for the DIL and MLC datasets, respectively.

Methods (using stand. split)	Performances	
Conv DBN [36]	99.2%	
Deep learning via embedding [37]	98.5%	
Deeply Supervised Nets [3]	<b>99.6%</b>	
<b>Our approach (↓)</b>	Baseline CNN	CoSen CNN
Stand. split (~600 trn, ~100 tst)	99.3%	99.3%
Low rep. (10%) of odd digits	97.6%	<b>98.6%</b>
Low rep. (10%) of even digits	97.1%	<b>98.4%</b>
Low rep. (25%) of odd digits	98.1%	<b>98.9%</b>
Low rep. (25%) of even digits	97.8%	<b>98.5%</b>

TABLE III: Evaluation on MNIST Database.

Methods (using stand. split)	Performances	
Network in Network [1]	64.3%	
Tree based Priors [38]	63.1%	
Probabilistic Maxout Network [39]	61.9%	
Maxout-Networks [40]	61.4%	
Stochastic Pooling [41]	57.5%	
Representation Learning [42]	60.8%	
Deeply Supervised Nets [3]	<b>65.4%</b>	
<b>Our approach (↓)</b>	Baseline CNN	CoSen CNN
Stand. split (500 trn, 100 tst)	65.2%	65.2%
Low rep. (10%) of odd digits	55.0%	<b>60.1%</b>
Low rep. (10%) of even digits	53.8%	<b>59.8%</b>
Low rep. (25%) of odd digits	57.7%	<b>61.5%</b>
Low rep. (25%) of even digits	57.4%	<b>61.6%</b>

TABLE IV: Evaluation on CIFAR-100 Database.

Our results for the two balanced datasets, MNIST and CIFAR-100, are reported in Tables III, IV on the standard splits along-with the deliberately imbalanced splits. To imbalance the training distributions, we used the available/normal training data for the even classes and only 25% and 10% of data for the odd classes. Similarly, we experimented by keeping the normal representation of the odd classes and reducing the representation of the even classes to only 25% and 10%. Our results show that the performance of our approach is equal to the performance of the baseline method when the distribution is balanced, but when the imbalance ratios increase, our approach produces significant improvements over the baseline CNN (which is trained without using the cost sensitive loss layer). We also compare with the state of the art techniques which report results on the standard split<sup>1</sup> and demonstrate that our performances are better or comparable. Note that for the MNIST digit dataset, nearly all the top performing approaches use distortions (affine and/or elastic) and data augmentation to achieve a significant boost in performance. In contrast, our baseline and cost sensitive CNNs do not use any form of distortions/augmentation during the training and testing procedures on MNIST.

We also experiment on the two popular classification datasets which are originally imbalanced, and for which the standard protocols use an equal number of images for all training classes. For example, 30 or 15 images are used for the case of Caltech-101 while 80 images per category are used in MIT-67 for training. We report our results on the standard splits (Tables V, VI), to compare with the state of

<sup>1</sup>Note that the standard split on the Caltech-101 and MIT-67 is different from the original data distribution (see Sec. IV-A for details).

Datasets		Performances						
(Imbalanced protocols)	Experimental Setting	Over-sampling (SMOTE [5])	Under-sampling (RUS [57])	Hybrid-sampling (SMOTE-RSB*[8])	CoSen SVM (WSVM [58])	CoSen RF (WRF [59])	Baseline CNN	CoSen CNN
MNIST	10% of odd classes	94.5%	92.1%	96.0%	96.8%	96.3%	97.6%	<b>98.6%</b>
CIFAR-100	10% of odd classes	32.2%	28.8%	37.5%	39.9%	39.0%	55.0%	<b>60.1%</b>
Caltech-101	60% trn, 10% of odd cl.	67.7%	61.4%	68.2%	70.1%	68.7%	77.4%	<b>83.2%</b>
MIT-67	60% trn, 10% of odd cl.	33.9%	28.4%	34.0%	35.5%	35.2%	50.4%	<b>56.9%</b>
DIL	stand. split (Exp. 2)	50.3%	46.7%	52.6%	55.3%	54.7%	69.5%	<b>72.6%</b>
MLC	stand. split (Exp. 2)	38.9%	31.4%	43.0%	47.7%	46.5%	66.1%	<b>68.6%</b>

TABLE VII: Comparisons of our approach with the state of the art class-imbalance approaches. The experimental protocols used for each dataset are shown in Fig. 7. With highly imbalanced training sets, our approach significantly out-performs other data sampling and cost sensitive classifiers on all four classification datasets.

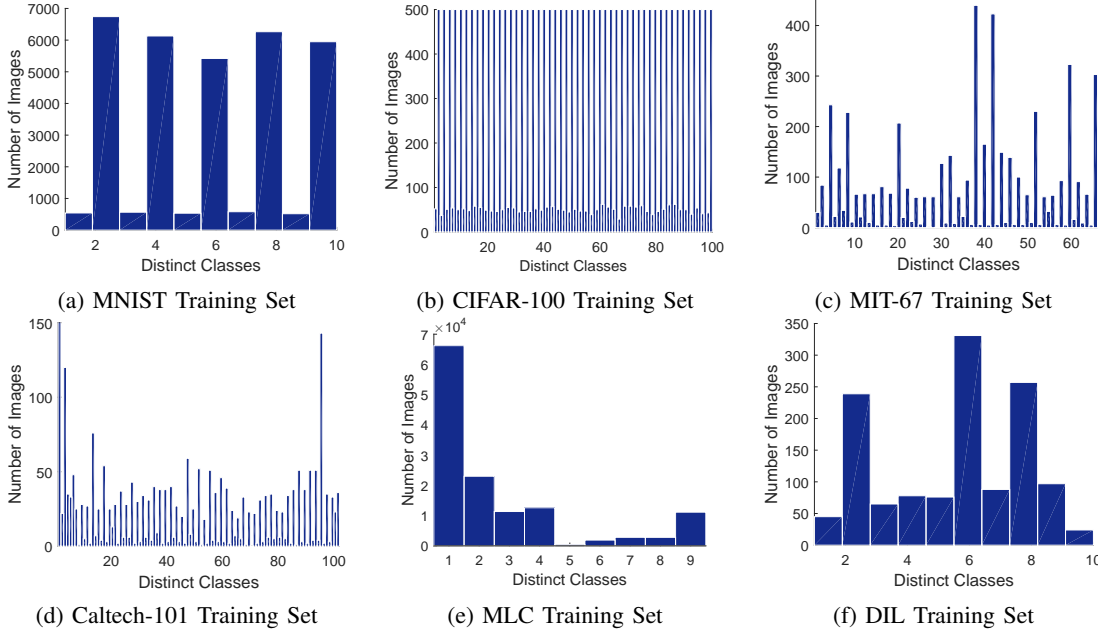


Fig. 7: The imbalanced training set distributions used for the comparisons reported in Table VII. Note that for the DIL and the MLC datasets, these distributions are the same as the standard protocols. For the MLC dataset, only the training set distribution for the first experiment is shown here which is very similar to the training set distribution of the second experiment (*best viewed when enlarged*).

Methods (using stand. split)	Performances	
	15 trn. samples	30 trn. samples
Multiple Kernels [43]	71.1 $\pm$ 0.6	78.2 $\pm$ 0.4
LLC <sup>†</sup> [44]	—	76.9 $\pm$ 0.4
Imp. Fisher Kernel <sup>†</sup> [45]	—	77.8 $\pm$ 0.6
SPM-SC [46]	73.2	84.3
DeCAF [47]	—	86.9 $\pm$ 0.7
Zeiler & Fergus [48]	83.8 $\pm$ 0.5	86.5 $\pm$ 0.5
Chatfield <i>et al.</i> [2]	—	88.3 $\pm$ 0.6
SPP-net [49]	—	<b>91.4 <math>\pm</math> 0.7</b>
Our approach ( $\downarrow$ )	Baseline CNN	CoSen CNN
Stand. split (15 trn. samples)	<b>87.1%</b>	<b>87.1%</b>
Stand. split (30 trn. samples)	90.8%	90.8%
Org. data distribution (60%/40% split)	88.1%	<b>89.3%</b>
Low rep. (10%) of odd classes	77.4%	<b>83.2%</b>
Low rep. (10%) of even classes	76.1%	<b>82.3%</b>
Org. data distribution (30%/70% split)	85.5%	<b>87.9%</b>
Low rep. (10%) of odd classes	74.6%	<b>80.4%</b>
Low rep. (10%) of even classes	75.2%	<b>80.9%</b>

TABLE V: Evaluation on Caltech-101 Database (<sup>†</sup> figures reported in [50]).

Methods (using stand. split)	Performances	
Spatial Pooling Regions [51]	50.1%	
VC + VQ [52]	52.3%	
CNN-SVM [53]	58.4%	
Improved Fisher Vectors [54]	60.8%	
Mid Level Representation [55]	64.0%	
Multiscale Orderless Pooling [56]	68.9%	
Our approach ( $\downarrow$ )	Baseline CNN	CoSen CNN
Stand. split (80 trn, 20 tst)	<b>70.9%</b>	<b>70.9%</b>
Org. data distribution (60%/40% split)	70.7%	<b>73.2%</b>
Low rep. (10%) of odd classes	50.4%	<b>56.9%</b>
Low rep. (10%) of even classes	50.1%	<b>56.4%</b>
Org. data distribution (30%/70% split)	61.9%	<b>66.2%</b>
Low rep. (10%) of odd classes	38.7%	<b>44.7%</b>
Low rep. (10%) of even classes	37.2%	<b>43.4%</b>

TABLE VI: Evaluation on MIT-67 Database.

the art approaches, and show that our results are superior to the state of the art on MIT-67 and competitive on the Caltech-101 dataset. Note that the best performing SPP-net [49] uses multiple sizes of Caltech-101 images during training. In contrast, we only use a single consistent size

Datasets	Performances			
	CoSen-CNN Fixed Cost (H)	CoSen-CNN Fixed Cost (S)	CoSen-CNN Fixed Cost (M)	CoSen-CNN Adap.
MNIST	97.2%	97.2%	97.9%	<b>98.6%</b>
CIFAR-100	55.2%	55.8%	56.0%	<b>60.1%</b>
Caltech-101	76.2%	77.1%	77.7%	<b>83.0%</b>
MIT-67	51.6%	50.9%	49.7%	<b>57.0%</b>
DIL	70.0%	69.5%	69.3%	<b>72.6%</b>
MLC	66.3%	66.8%	65.7%	<b>68.6%</b>

TABLE VIII: Comparisons of our approach (adaptive costs) with the fixed class-specific costs. The experimental protocols used for each dataset are shown in Fig. 7. Fixed costs do not show a significant and consistent improvement in results.

during training and testing. We also experiment with the original imbalanced data distributions to train the CNN with the modified loss function. For the original data distributions, we use both 60%/40% and 30%/70% train/test splits to show our performances with a variety of train/test distributions. Moreover, with these imbalanced splits, we further decrease the data of odd and even classes to just 10% respectively, and observe a better relative performance of our proposed approach compared to the baseline method.

The comparisons with the best approaches for class-imbalance learning are shown in Table VII. Note that we used a high degree of imbalance for the case of all six datasets to clearly show the impact of the class imbalance problem on the performance of the different approaches (Fig.7). For fairness and conclusive comparisons, our experimental procedure was kept as close as possible to the proposed CoSen CNN. For example, for the case of CoSen Support Vector Machine (SVM) and Random Forest (RF) classifiers, we used the 4096 dimensional features extracted from the pre-trained deep CNN (D) [34]. Similarly, for the cases of over and under-sampling, we used the same 4096 dimensional features, which have shown to perform well on other classification datasets. We also report comparisons with all types of data sampling techniques i.e., over-sampling (SMOTE [5]), under-sampling (Random Under Sampling - RUS [57]) and hybrid sampling (SMOTE-RSB\* [8]). Note that despite the simplicity of the approaches in [5, 57], they have been shown to perform very well on imbalanced datasets in data mining [15, 60]. We also compare with the cost sensitive versions of popular classifiers (weighted SVM [58] and weighted RF [59]). For the case of weighted SVM, we used the standard implementation of LIBSVM [61] and set the class dependent costs based on the proportion of each class in the training set. Our proposed approach demonstrates a significant improvement over all of the cost sensitive class imbalance methods.

Since our approach updates the costs with respect to the data statistics (i.e., data distribution, class separability and classification errors), an interesting aspect is to analyse the performance when the costs are fixed and set equal to these statistics instead of updating them adaptively. We experiment with fixed costs instead of adaptive costs in the case of CoSen-CNN. For this purpose, we used three versions of fixed costs, based on the class representation (H), data separability (S) and classification errors (M). Table VIII shows the results for

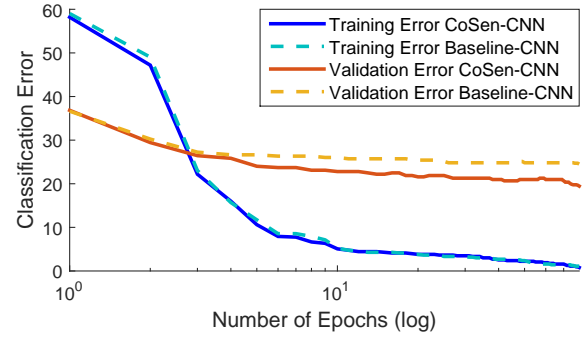


Fig. 8: An observed decrease in the training and validation error on the DIL dataset (stand. split, Exp. 2) for the cases of the baseline and cost-sensitive CNNs.

each dataset with four different types of costs. The results show that none of the fixed costs significantly improve the performance in comparison to the adaptive cost. This shows that the optimal costs are not the H, S and M themselves, rather an intermediate set of values give the best performance for cost-sensitive learning.

**Timing Comparisons:** The introduction of the class dependent costs did not prove to be prohibitive during the training of the CNN. For example, on an Intel quad core i7-4770 CPU (3.4GHz) with 32Gb RAM and Nvidia GeForce GTX 660 card (2GB), it took 80.19 secs and 71.87 secs to run one epoch with and without class sensitive parameters, respectively for the MIT-67 dataset. At test time, the CoSen CNN took the same amount of time as that of the baseline CNN, because no extra computations were involved during testing.

## V. CONCLUSION

We proposed a cost sensitive deep CNN to deal with the class-imbalance problem, which is commonly encountered when dealing with real world datasets. Our approach is able to automatically set the class dependent costs based on the data statistics of the training set. We analysed three commonly used cost functions and introduced class dependent costs for each case. We show that the cost-sensitive CE loss function is c-calibrated and guess averse. Furthermore, we proposed an alternating optimization procedure to efficiently learn the class dependent costs as well as the network parameters. Our results on six popular classification datasets show that the modified cost functions perform very well on the majority as well as on the minority classes in the dataset.

## ACKNOWLEDGMENTS

This research was supported by an IPRS awarded by The University of Western Australia and Australian Research Council (ARC) grants DP150100294 and DE120102960.

## APPENDIX A

### PROOFS REGARDING COST MATRIX $\xi'$

**Lemma A.1.** *Offsetting the columns of the cost matrix  $\xi'$  by any constant 'c' does not affect the associated classification risk  $\mathcal{R}$ .*

*Proof:* From Eq. 1, we have:

$$\sum_q \xi'_{p^*,q} P(q|\mathbf{x}) \leq \sum_q \xi'_{p,q} P(q|\mathbf{x}) \quad \forall p \neq p^*$$

which gives the following relation:

$$\sum_{q \neq p^*} P(q|\mathbf{x}) (\xi'_{p^*,q} - \xi'_{p,p^*}) \leq 0, \quad \forall p \neq p^*$$

As indicated in Sec. 3.1, the above expression holds for all  $p \neq p^*$ . For a total number of  $N$  classes and an optimal prediction  $p^*$ , there are  $N - 1$  of the above relations. By adding up the left and the right hand sides of these  $N - 1$  relations we get:

$$P(p^*|\mathbf{x}) \left( (N-1)\xi'_{p^*,p^*} - \sum_{p \neq p^*} \xi'_{p,p^*} \right) \leq \sum_{q \neq p^*} P(q|\mathbf{x}) \left( \sum_{p \neq p^*} \xi'_{p,q} - (N-1)\xi'_{p^*,q} \right),$$

This can be simplified to:

$$\mathbf{P}_{\mathbf{x}} \begin{bmatrix} \sum_i \xi'_{i,1} - N\xi'_{p^*,1} \\ \vdots \\ \sum_i \xi'_{i,N} - N\xi'_{p^*,N} \end{bmatrix} \geq 0,$$

where,  $\mathbf{P}_{\mathbf{x}} = [P(1|\mathbf{x}), \dots, P(N|\mathbf{x})]$ . Note that the posterior probabilities  $\mathbf{P}_{\mathbf{x}}$  are positive ( $\sum_i P(i|\mathbf{x}) = 1$  and  $P(i|\mathbf{x}) > 0$ ). It can be seen from the above equation that the addition of any constant  $c$ , does not affect the overall relation, i.e., for any column  $j$ ,

$$\sum_i (\xi'_{i,j} + c) - N(\xi'_{p^*,j} + c) = \sum_i \xi'_{i,j} - N\xi'_{p^*,j}$$

Therefore, the columns of the cost matrix can be shifted by a constant  $c$  without any effect on the associated risk. ■

**Lemma A.2.** *The cost of the true class should be less than the mean cost of all misclassification.*

*Proof:* Since,  $\mathbf{P}_{\mathbf{x}}$  can take any distribution of values, we end up with the following constraint:

$$\sum_i \xi'_{i,j} - N\xi'_{p^*,j} \geq 0, \quad j \in [1, N].$$

For a correct prediction  $p^*$ ,  $P(p^*|\mathbf{x}) > P(p|\mathbf{x}), \forall p \neq p^*$ . Which implies that:

$$\xi'_{p^*,p^*} \leq \frac{1}{N} \sum_i \xi'_{i,p^*}.$$

It can be seen that the cost insensitive matrix (when  $\text{diag}(\xi') = 0$  and  $\xi'_{i,j} = 1, \forall j \neq i$ ) satisfies this relation and provides the upper bound. ■

## REFERENCES

- [1] M. Lin, Q. Chen, and S. Yan, "Network in network," in *Statistical Language and Speech Processing*, 2013.
- [2] K. Chatfield, K. Simonyan, A. Vedaldi, and A. Zisserman, "Return of the devil in the details: Delving deep into convolutional nets," *arXiv:1405.3531*, 2014.
- [3] C.-Y. Lee, S. Xie, P. Gallagher, Z. Zhang, and Z. Tu, "Deeply-supervised nets," in *AISTATS*, 2015.
- [4] O. Beijbom, P. J. Edmunds, D. Kline, B. G. Mitchell, D. Kriegman *et al.*, "Automated annotation of coral reef survey images," in *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*. IEEE, 2012, pp. 1170–1177.
- [5] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "Smote: synthetic minority over-sampling technique," *JAIR*, vol. 16, no. 1, pp. 321–357, 2002.
- [6] S. H. Khan, M. Bennamoun, F. Sohel, and R. Togneri, "Automatic feature learning for robust shadow detection," in *CVPR*. IEEE, 2014, pp. 1939–1946.
- [7] Y. Zhang and D. Wang, "A cost-sensitive ensemble method for class-imbalanced datasets," in *AAA*, vol. 2013. Hindawi, 2013.
- [8] E. Ramentol, Y. Caballero, R. Bello, and F. Herrera, "Smote-rsb\*: a hybrid preprocessing approach based on oversampling and undersampling for high imbalanced data-sets using smote and rough sets theory," *KIS*, vol. 33, no. 2, pp. 245–265, 2012.
- [9] M. Kukar, I. Kononenko *et al.*, "Cost-sensitive learning with neural networks," in *ECAI*. Citeseer, 1998, pp. 445–449.
- [10] Z.-H. Zhou and X.-Y. Liu, "Training cost-sensitive neural networks with methods addressing the class imbalance problem," *Knowledge and Data Engineering, IEEE Transactions on*, vol. 18, no. 1, pp. 63–77, 2006.
- [11] S. H. Khan, M. Hayat, M. Bennamoun, R. Togneri, and F. Sohel, "A discriminative representation of convolutional features for indoor scene recognition," *arXiv preprint arXiv:1506.05196*, 2015.
- [12] L. Maaten, M. Chen, S. Tyree, and K. Q. Weinberger, "Learning with marginalized corrupted features," in *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, 2013, pp. 410–418.
- [13] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *NIPS*, 2012, pp. 1097–1105.
- [14] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.
- [15] V. Garcia, J. Sanchez, J. Mollineda, R. Alejo, and J. Sotoca, "The class imbalance problem in pattern classification and learning," in *Congreso Espanol de Informatica*, 2007, pp. 1939–1946.
- [16] K.-J. Wang, B. Makond, K.-H. Chen, and K.-M. Wang, "A hybrid classifier combining smote with pso to estimate 5-year survivability of breast cancer patients," *ASC*, vol. 20, pp. 15–24, 2014.
- [17] H. Han, W.-Y. Wang, and B.-H. Mao, "Borderline-smote: a new over-sampling method in imbalanced data sets learning," in *AIC*. Springer, 2005, pp. 878–887.
- [18] C. Bunkhumpornpat, K. Sinapiromsaran, and C. Lursinsap, "Safe-level-smote: Safe-level-synthetic minority over-sampling technique for handling the class imbalanced problem," in *AKDDM*. Springer, 2009, pp. 475–482.
- [19] T. Maciejewski and J. Stefanowski, "Local neighbourhood extension of smote for mining imbalanced data," in *CIDM*. IEEE, 2011, pp. 104–111.
- [20] G. E. Batista, R. C. Prati, and M. C. Monard, "A study of the behavior of several methods for balancing machine learning training data," *ACM Sigkdd Explorations Newsletter*, vol. 6, no. 1, pp. 20–29, 2004.
- [21] P. Jeatrakul, K. W. Wong, and C. C. Fung, "Classification of imbalanced data by combining the complementary neural network and smote algorithm," in *NIP*. Springer, 2010, pp. 152–159.
- [22] M. Gao, X. Hong, and C. J. Harris, "Construction of neurofuzzy models for imbalanced data classification," *TFS*, vol. 22, no. 6, pp. 1472–1488, 2014.
- [23] Y. Zhang, P. Fu, W. Liu, and G. Chen, "Imbalanced data classification based on scaling kernel-based support vector machine," *NCA*, vol. 25, no. 3–4, pp. 927–935, 2014.
- [24] K. Li, X. Kong, Z. Lu, L. Wenyan, and J. Yin, "Boosting weighted elm for imbalanced learning," *Neurocomputing*, vol. 128, pp. 15–21, 2014.
- [25] B. X. Wang and N. Japkowicz, "Boosting support vector machines for imbalanced data sets," *KIS*, vol. 25, no. 1, pp. 1–20, 2010.
- [26] G. Wu and E. Y. Chang, "Class-boundary alignment for imbalanced dataset learning," in *ICML workshop*, 2003, pp. 49–56.
- [27] —, "Kba: Kernel boundary alignment considering imbalanced data distribution," *TKDE*, vol. 17, no. 6, pp. 786–795, 2005.

- [28] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*. IEEE, 2014, pp. 580–587.
- [29] K. Huang, H. Yang, I. King, and M. R. Lyu, "Learning classifiers from imbalanced data based on biased minimax probability machine," in *CVPR*, vol. 2. IEEE, 2004, pp. II–558.
- [30] P. L. Bartlett, M. I. Jordan, and J. D. McAuliffe, "Convexity, classification, and risk bounds," *Journal of the American Statistical Association*, vol. 101, no. 473, pp. 138–156, 2006.
- [31] C.-Y. Lee, S. Xie, P. Gallagher, Z. Zhang, and Z. Tu, "Deeply-supervised nets," *arXiv:1409.5185*, 2014.
- [32] M. A. Nielsen, "Neural networks and deep learning," *Determination Press*, vol. 1, 2014.
- [33] L. Ballerini, R. B. Fisher, B. Aldridge, and J. Rees, "A color and texture based hierarchical k-nn approach to the classification of non-melanoma skin lesions," in *Color Medical Image Analysis*. Springer, 2013, pp. 63–86.
- [34] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv:1409.1556*, 2014.
- [35] L. Ballerini, R. B. Fisher, B. Aldridge, and J. Rees, "Non-melanoma skin lesion classification using colour image data in a hierarchical k-nn classifier," in *Biomedical Imaging (ISBI), 2012 9th IEEE International Symposium on*. IEEE, 2012, pp. 358–361.
- [36] H. Lee, R. Grosse, R. Ranganath, and A. Y. Ng, "Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations," in *ICML*. ACM, 2009, pp. 609–616.
- [37] J. Weston, F. Ratle, H. Mobahi, and R. Collobert, "Deep learning via semi-supervised embedding," in *Neural Networks: Tricks of the Trade*. Springer, 2012, pp. 639–655.
- [38] N. Srivastava and R. R. Salakhutdinov, "Discriminative transfer learning with tree-based priors," in *NIPS*, 2013, pp. 2094–2102.
- [39] J. T. Springenberg and M. Riedmiller, "Improving deep neural networks with probabilistic maxout units," *ICLRs*, 2014.
- [40] I. Goodfellow, D. Warde-Farley, M. Mirza, A. Courville, and Y. Bengio, "Maxout networks," in *ICML*, 2013, pp. 1319–1327.
- [41] M. D. Zeiler and R. Fergus, "Stochastic pooling for regularization of deep convolutional neural networks," *arXiv:1301.3557*, 2013.
- [42] T.-H. Lin and H. Kung, "Stable and efficient representation learning with nonnegativity constraints," in *ICML*, 2014, pp. 1323–1331.
- [43] A. Vedaldi, V. Gulshan, M. Varma, and A. Zisserman, "Multiple kernels for object detection," in *ICCV*. IEEE, 2009, pp. 606–613.
- [44] J. Wang, J. Yang, K. Yu, F. Lv, T. Huang, and Y. Gong, "Locality-constrained linear coding for image classification," in *CVPR*. IEEE, 2010, pp. 3360–3367.
- [45] F. Perronnin, J. Sánchez, and T. Mensink, "Improving the fisher kernel for large-scale image classification," in *ECCV*. Springer, 2010, pp. 143–156.
- [46] J. Yang, K. Yu, Y. Gong, and T. Huang, "Linear spatial pyramid matching using sparse coding for image classification," in *CVPR*. IEEE, 2009, pp. 1794–1801.
- [47] J. Donahue, Y. Jia, O. Vinyals, J. Hoffman, N. Zhang, E. Tzeng, and T. Darrell, "Decaf: A deep convolutional activation feature for generic visual recognition," in *ICML*, 2014, pp. 647–655.
- [48] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *ECCV*. Springer, 2014, pp. 818–833.
- [49] K. He, X. Zhang, S. Ren, and J. Sun, "Spatial pyramid pooling in deep convolutional networks for visual recognition," *arXiv:1406.4729*, 2014.
- [50] K. Chatfield, V. Lempitsky, A. Vedaldi, and A. Zisserman, "The devil is in the details: an evaluation of recent feature encoding methods," 2011.
- [51] D. Lin, C. Lu, R. Liao, and J. Jia, "Learning important spatial pooling regions for scene classification," 2014.
- [52] Q. Li, J. Wu, and Z. Tu, "Harvesting mid-level visual concepts from large-scale internet images," in *CVPR*. IEEE, 2013, pp. 851–858.
- [53] A. S. Razavian, H. Azizpour, J. Sullivan, and S. Carlsson, "Cnn features off-the-shelf: an astounding baseline for recognition," in *CVPRw*. IEEE, 2014, pp. 512–519.
- [54] M. Juneja, A. Vedaldi, C. Jawahar, and A. Zisserman, "Blocks that shout: Distinctive parts for scene classification," in *CVPR*. IEEE, 2013, pp. 923–930.
- [55] C. Doersch, A. Gupta, and A. A. Efros, "Mid-level visual element discovery as discriminative mode seeking," in *NIPS*, 2013, pp. 494–502.
- [56] Y. Gong, L. Wang, R. Guo, and S. Lazebnik, "Multi-scale orderless pooling of deep convolutional activation features," in *ECCV*. Springer International Publishing, 2014, pp. 392–407.
- [57] I. Mani and I. Zhang, "knn approach to unbalanced data distributions: a case study involving information extraction," in *WLID*, 2003.
- [58] Y. Tang, Y.-Q. Zhang, N. V. Chawla, and S. Krasser, "Svms modeling for highly imbalanced classification," *TSMC*, vol. 39, no. 1, pp. 281–288, 2009.
- [59] C. Chen, A. Liaw, and L. Breiman, "Using random forest to learn imbalanced data," *University of California, Berkeley*, 2004.
- [60] H. He and E. A. Garcia, "Learning from imbalanced data," *TKDE*, vol. 21, no. 9, pp. 1263–1284, 2009.
- [61] C.-C. Chang and C.-J. Lin, "Libsvm: a library for support vector machines," *TIST*, vol. 2, no. 3, p. 27, 2011.