Learning from Imbalanced Datasets with Cross-View Cooperation-Based Ensemble Methods

Cécile Capponi, Sokol Koço

Learning from imbalanced datasets with cross-view cooperation-based ensemble methods*

Cécile Capponi and Sokol Koço

Aix-Marseille Université, Université de Toulon, CNRS, LIS, Marseille, France
cecile.capponi@lis-lab.fr

Abstract. In this paper, we address the problem of learning from imbalanced multi-class datasets in a supervised setting when multiple descriptions of the data – also called views – are available. Each view incorporates various information on the examples, and in particular, depending on the task at hand, each view might be better at recognizing only a subset of the classes. Establishing a sort-of cooperation between the views is needed for all the classes to be equally recognized — a crucial problem particularly for imbalanced datasets. The novelty of our work consists in capitalizing on the complementariness of the views so that each class can be processed by the most appropriate view(s); thus improving the per-class performances of the final classifier. The main contribution of this paper are two ensemble learning methods based on recent theoretical works on the use of the confusion matrix's norm as an error measure, while empirical results show the benefits of the proposed approaches.

Keywords: ensemble methods, imbalanced classes, confusion matrix norm, boosting, multi-view learning

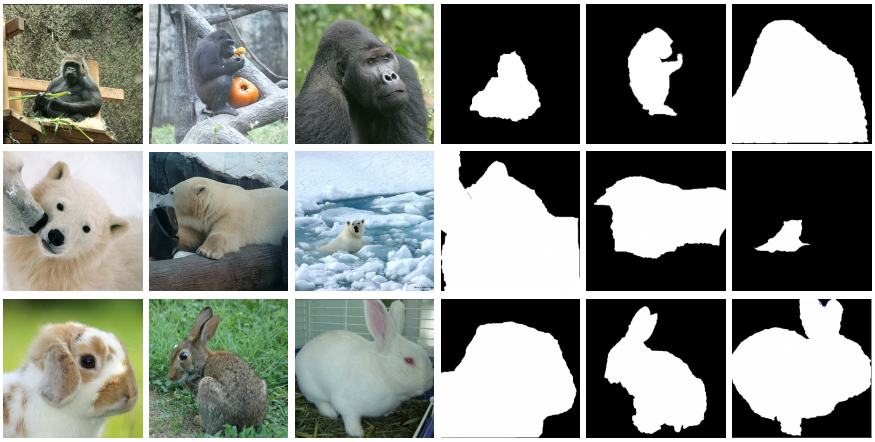
1 Introduction

In machine learning, a frequent issue for datasets coming from real-life applications is the problem of imbalanced classes: some classes (called *majority* classes) are more represented than the others (the *minority* ones). On the other hand, the data may be described by different sets of attributes, also called *views*. The capability of views to deal with a multi-class learning problem is uneven: some views are usually more appropriate than others to process some classes (*cf.* Fig. 1) : it is therefore worthwhile to encourage the right views to recognize the right classes, especially in the case of classes that are underpopulated.

The purpose of the work presented in this paper is to address these two imbalanced properties as one: *how to exploit the specificities that each view has on a per class basis so that the final classifier equally recognizes both majority and minority classes?*

To the best of our knowledge, multi-view and imbalanced multi-class supervised classification problems have always been tackled in separate ways. On one

* This work is partially funded by the French ANR project LIVES ANR-15-CE23-0026



(a) The original images (the first view is extracted from them, focusing on colors)

(b) The animals are separated from the background (the second view)

Fig. 1: Extract from the dataset *Animals with Attributes* [14]: three examples of classes Gorilla, Polar bear, and Rabbit. One first view reflecting colors (such as histograms of colors) would be sufficient to distinguish polar bears from gorillas, but not enough to discriminate either gorillas from rabbits or rabbits from polar bears. For achieving these two lattest purposes, one might expect than images descriptors reflecting edges or segmentation would be better than colors. Multi-class and multi-view classifications are sometimes entangled.

side, the supervised multi-view setting is often processed through early fusion of description spaces, or through late fusion of every classifier learnt from the various description spaces [18, 1]. Among the late fusion approaches, the indisputable success of MKL [2] must be balanced with the time required for its processing of high input space dimensions [8]. On the other side, the imbalanced classes problems have been addressed through two main approaches [10, 9]:

- resampling (e.g. SMOTE: [4]) which aims at rebalancing the sample,
- cost-sensitive methods which make use of class-based loss functions for building models that take into account the imbalanced rate of the training set (e.g.: [21]).

Both approaches are sometimes coupled with specific feature selection techniques (e.g.: [23]) such that the most appropriate features are identified for recognizing the hardest (smallest) classes.

In this paper, we advocate that promoting the cooperation between imbalanced views leads to finding the most appropriate view(s) for each class, thus improving the performances of the final classifier for imbalanced classes problems. Fig. 2 depicts an example of distributed confusion between classes among views according to the Bayes error: such a problem arises in many balanced

datasets, and is naturally amplified with class imbalance due to the $P(y)$ of the Bayes' rule[1]. We propose here an algorithm that encourages the cooperation among views/classes in order to select the right view for reducing the confusion between each couple of classes.

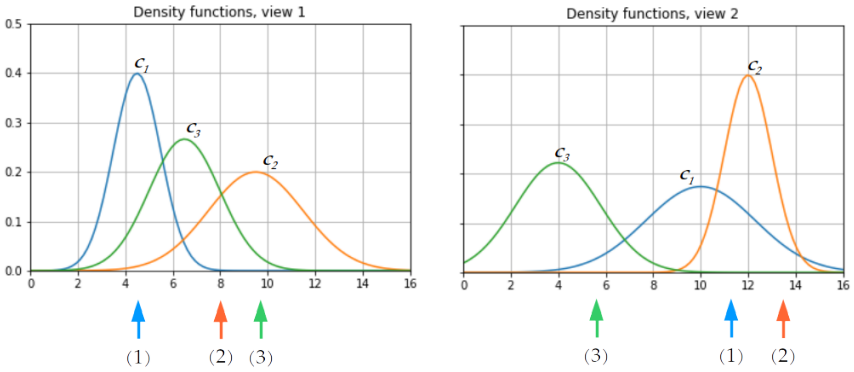


Fig. 2: Multi-view multi-class imbalanced confusions in a nutshell. Here are density functions of three classes among two views: the description space is $\mathbb{R}$ in both views ($\mathcal{X} = \mathbb{R} \times \mathbb{R}$), and 3 examples are pictured according to their description values in each view. Independently from class imbalance (which is not represented), one can easily notice that the confusion rates according to the Bayes error are not balanced within both views: for example there is more confusion between c_2 and c_3 in view 1 than in view 2. If the Bayes rule would apply on the first view, then the example 3 would be classified in c_2, whereas on the second view the decision would be c_3. This illustrates that both views should be considered and combined in a proper way in order to decide the right class for each example's data. If we consider the class imbalance problem (which means that $P(y)$ would favor classes over-represented in the dataset), then example 2 is more likely to be classified as c_3 rather than c_2 in view 1 as far as c_2 is under-represented compared to c_3, whereas that confusion is less proeminent in view 1.

Based on recent theoretical results on the use of the confusion matrix's norm as an error measure (e.g. [17]), the aim of the proposed methods is to find the best combination of views for each class ensuring a small confusion norm. Roughly speaking, our proposal is a cost-sensitive method combined with a greedy selection among predefined groups of features (named views).

The remainder of this paper is organized as follows: Sections 2 and 3 introduce the notation used in this paper and the motivations and frameworks on which the proposed approaches rely on. The main contribution of this paper is given in Sections 4 and 5, where we show step by step how to derive multi-view methods for the imbalanced setting. Finally, Section 6 gives the experimental results, and we conclude in Sections 7 and 8.

[1] The Bayes rule $B(x)$ chooses y that maximizes $P(x|y)P(y)/P(x)$.

2 Theoretical inspiration from multi-class boosting: settings

2.1 General notation

Matrices and vectors are denoted by bold capital letters like $\mathbf{C}$ and bold small letters like $\mathbf{x}$ respectively. The entry of the l-th row and the k-th column of $\mathbf{C}$ is denoted $\mathbf{C}(l,k)$, or simply $\mathbf{c}_{l,k}$. $\lambda_{max}(\mathbf{C})$ and $Tr(\mathbf{C})$ respectively correspond to the largest eigenvalue and the trace of $\mathbf{C}$. The *spectral* or *operator norm* $\|\mathbf{C}\|$ of $\mathbf{C}$ is defined as:

$$\|\mathbf{C}\| \stackrel{\text{def}}{=} \max_{\mathbf{v}\neq 0} \frac{\|\mathbf{C}\mathbf{v}\|_2}{\|\mathbf{v}\|_2} \stackrel{\text{def}}{=} \sqrt{\lambda_{max}(\mathbf{C}^*\mathbf{C})},$$

where $\|\cdot\|_2$ is the Euclidian norm and $\mathbf{C}^*$ is the conjugate transpose of $\mathbf{C}$. The inner product and the Frobenius inner product of two matrices $\mathbf{A}$ and $\mathbf{B}$ are denoted $\mathbf{AB}$ and $\mathbf{A}\cdot\mathbf{B}$, respectively.

The indicator function is denoted by $\mathbb{I}$; K is the number of classes, m the number of views, n the number of examples, and n_y the number of examples of class y, where $y \in \{1,...,K\}$. X, Y and $\mathcal{H}$ are the input, output and hypothesis spaces, respectively; $(x_i, y_i), x_i$ or i are interchangeably used to denote the i^{th} training example.

2.2 Multi-class boosting framework

In this paper we use the boosting framework for multi-class classification introduced in [16], and more precisely the one defined for AdaBoost.MM. Algorithms based on the AdaBoost family maintain a distribution over the training samples in order to identify hard-to-classify examples: the greater the weight of an example, the greater the need to correctly classify these data. In the considered setting, the distribution over the training examples is replaced by a cost matrix. Let $S = \{(x_i, y_i)\}_{i=1}^n$ be a training sample, where $x_i \in X$ and $y_i \in \{1,...,K\}$. The cost matrix $\mathbf{D} \in \mathbb{R}^{n\times K}$ is constructed so that for a given example (x_i, y_i), $\forall k \neq y_i$: $\mathbf{D}(i, y_i) \leq \mathbf{D}(i,k)$, where i is the row of $\mathbf{D}$ corresponding to (x_i, y_i).

This cost matrix is a particular case of cost matrices used in cost sensitive methods (for example, [20]), where classification costs are given for each example and each class. However, contrary to those methods, the matrix is not given prior to the learning process, but it is updated after each iteration of AdaBoost.MM so that the misclassification cost reflects the difficulty of correctly classifying an example. That is, the costs are increased for examples that are hard to classify, and they are decreased for easier ones.

In AdaBoost.MM, the cost matrix $\mathbf{D}$ at iteration T, is defined as follows:

$$\mathbf{D}_T(i,k) \stackrel{\text{def}}{=} \begin{cases} e^{f_T(i,k)-f_T(i,y_i)} & \text{if } k \neq y_i \\ -\sum\limits_{k\neq y_i} e^{f_T(i,k)-f_T(i,y_i)} & \text{otherwise,} \end{cases}$$

where $f_T(i,k)$ is the score function computed as: $f_T(i,k) = \sum_{t=1}^T \alpha_t \mathbb{I}[h_t(i) = k]$.

At each iteration t, AdaBoost.MM selects the classifier h and its weight α that minimize the exponential loss:

$$h_t, \alpha_t = \underset{h,\alpha}{\operatorname{argmin}} \sum_{i=1}^{n} \sum_{k \neq y_i} e^{f_t(i,k) - f_t(i,y_i)}, \tag{1}$$

$$\text{where: } f_t(i,k) = \sum_{s=1}^{t-1} \alpha_s \mathbb{I}[h_s(i) = k] + \alpha \mathbb{I}[h(i) = k].$$

The final hypothesis of AdaBoost.MM is a weighted majority vote:

$$H(x) = \underset{k=1\ldots K}{\operatorname{argmax}} \; f_T(i,k).$$

2.3 The supervised multi-view setting

Following previous works [13, 12, 11], the multi-view setting that we considered in this paper is supervised. In the case of mono-view classification, where a predictor capable of reliably computing the label associated with some input data, the learning problem hinges on a training set $S = \{(x_i, y_i)\}_{i=1}^{n}$, which is an i.i.d. sample of n observations where $x_i \in \mathcal{X}$, $y_i \in \mathcal{Y}$, and $(x_i, y_i) \sim \mathcal{D}_{\mathcal{X}\times\mathcal{Y}} = \mathcal{D}$. In the multi-view setting, the input space $\mathcal{X}$ is a product of m spaces $\mathcal{X}^{(v)}$, $v = 1..m$, and we note $\mathcal{D}^{(v)} \sim P(X^{(v)}, Y)$ where $X^{(v)}$ is a random variable that ranges in $\mathcal{X}^{(v)}$.

Multi-view learning is the usual problem of learning $\hat{h} \in \mathcal{H} : \mathcal{X} \to \mathcal{Y}$ such that the generalisation risk is minimized, *i.e.*

$$\hat{h} = \operatorname{argmin}_{h\in\mathcal{H}} \mathcal{R}(h) = \operatorname{argmin}_{h\in\mathcal{H}} \mathbb{E}_{\sim\mathcal{D}} \left[\ell(h(x), y)\right]$$

where $\ell : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}^+$ is some loss function, for example the 0-1 loss: $\ell(y, y') = \mathbf{1}_{y\neq y'}$.

For each view v, let $\hat{h}^{(v)} : \mathcal{X}^{(v)} \to \mathcal{Y}$ be such that

$$\hat{h}^{(v)} = \operatorname{argmin}_{h\in H^{(v)}} \mathbb{E}_{\sim\mathcal{D}^{(v)}} \left[\ell(h^{(v)}(x), y)\right]$$

In order for the multiview learning to be worthwhile, we obviously expect that $\forall v \in [1..m]$,

$$\mathbb{E}_{\sim\mathcal{D}^{(v)}} \left[\ell(\hat{h}^{(v)}(x), y)\right] > \mathbb{E}_{\sim\mathcal{D}} \left[\ell(\hat{h}(x), y)\right]$$

which means that it is beneficial to consider several views than the best one.

3 Reducing the class confusion with multi-view insights

3.1 Confusion matrix: a probabilistic definition

When dealing with imbalanced classes, a common tool used to measure the goodness of a classifier is the confusion matrix. Previous works on the confusion

matrix (e.g. [15, 17], etc.), advocate that using the operator norm of (a particular formulation of) the confusion matrix as an optimization criterion is a reasonable choice for dealing with the imbalanced classes problem. Following in their steps, in this paper we consider a particular definition of the confusion matrix:

Definition 1. *For a given classifier $H \in \mathcal{H}$, an unknown distribution $\mathcal{D}$ over $X \times Y$, and a training set $S = \{(x_i, y_i)\}_{i=1}^n$ i.i.d according to $\mathcal{D}$, the true and empirical confusion matrices of h, denoted $\mathbf{C} = (\mathbf{c}_{l,k})_{1 \leq l,k \leq K}$ and $\mathbf{C}_S = (\hat{\mathbf{c}}_{l,k})_{1 \leq l,k \leq K}$ respectively, are defined as:*

$$\mathbf{c}_{l,k} \overset{def}{=} \begin{cases} 0 & \text{if } l = k \\ \mathbb{P}_{(x,y)\sim\mathcal{D}}(H(x) = k | y = l) & \text{otherwise.} \end{cases}$$

$$\hat{\mathbf{c}}_{l,k} \overset{def}{=} \begin{cases} 0 & \text{if } l = k \\ \sum_{i=1}^{n} \frac{1}{n_l} \mathbb{I}[H(x_i) = k] \mathbb{I}[y_i = l] & \text{otherwise,} \end{cases}$$

Contrary to the usual (probabilistic) definition of the confusion matrix, the diagonal entries are zeroed. The advantage of this formulation is two fold: first, it takes into account only the errors of the classifier, and second, its operator norm gives a bound on the true risk of the classifier. Indeed, let $\mathbf{p} = [P(y = 1), ..., P(y = K)]$ be the vector of class priors distribution, then we have:

$$R(h) \overset{\text{def}}{=} P_{(x,y)\sim\mathcal{D}}(H(x) \neq y) = \|\mathbf{pC}\|_1 \leq \sqrt{K}\|\mathbf{C}\|, \tag{2}$$

where $R(h)$ is the true risk of h and $\|\cdot\|_1$ denotes the $l1$-norm. The aim of the methods presented is this paper is thus to find a classifier $\hat{H}$ that verifies the following criterion:

$$\hat{H} = \underset{H \in \mathcal{H}}{\operatorname{argmin}} \|\mathbf{C}\|. \tag{3}$$

3.2 From the confusion matrix norm to an optimization problem

The confusion matrix $\mathbf{C}$, as given in Definition 1, depends on the unknown distribution $\mathcal{D}$, thus making the problem given in Equation 3 difficult to tackle directly. In order to bypass this, we make use of Theorem 1 in [17], which bounds the norm of the *true* confusion matrix by the norm of its empirical estimation. We give here a reformulation of the theorem for the supervised setting, where the considered loss is the indicator function $\mathbb{I}$.

Corollary 1 *For any $\delta \in (0; 1]$, it holds with probability $1 - \delta$ over a sample $S_{(\mathbf{x},y)\sim\mathcal{D}}$ that:*

$$\|\mathbf{C}\| \leq \|\mathbf{C}_S\| + \sqrt{2K \sum_{k=1}^{K} \frac{1}{n_k} \log \frac{K}{\delta}},$$

where $\mathbf{C}_S$ is the empirical confusion matrix computed for a classifier h over S.

The direct implication of Corollary 1 is that minimizing the norm of the empirical confusion matrix results in the minimization of its true norm. Unfortunately, due to the nature of the confusion matrix in Definition 1, an analytical expression for the norm of the matrix is difficult to compute. We thus consider an upper bound on $\|\mathbf{C}_S\|^2$:

$$\|\mathbf{C}_S\|^2 = \lambda_{max}(\mathbf{C}_S^*\mathbf{C}_S) \leq Tr(\mathbf{C}_S^*\mathbf{C}_S) \leq \sum_{l=1}^{K}\sum_{k\neq l}\hat{\mathbf{c}}_{l,k} = \|\mathbf{C}_S\|_1. \tag{4}$$

In the last part of Equation 4, we abuse the notation and denote the entry-wise $l1$-norm of the matrix by $\|\mathbf{C}\|_1$. Equation 4 implies that the updated goal is:

$$\hat{H} = \underset{H\in\mathcal{H}}{\operatorname{argmin}}\|\mathbf{C}\|_1. \tag{5}$$

Another drawback of the confusion matrix as given in Definition 1 is the presence of the indicator function, which is not optimization friendly. One way to handle this is to replace the indicator function with loss functions $\ell_{l,k}(H,x)$ defined over two classes, so that $\forall(x,y)\in S$ and $l,k\in\{1\dots K\}$, $\mathbb{I}(h(x)\neq y)\leq \ell_{l,k}(H,x)$. Applying these losses to Equation 4, we have the actual upper bound of the confusion matrix's norm:

$$\|\mathbf{C}\|_1 \leq \sum_{i=1}^{n}\sum_{k\neq y_i}\frac{1}{n_{y_i}}\ell_{y_i,k}(H,\mathbf{x}_i). \tag{6}$$

3.3 A multi-view classifier

In the multi-view setting, an example is represented by various sets of attributes, called views. Although they represent the same objects, different views might be suited for different tasks and/or categories (classes) of objects. For instance, in image classification, the view *color* is more suited for distinguishing between gorillas and polar bears, than between cats and dogs. The motivation behind the work in this paper is to deal with both *imbalanced data* (some classes are more represented than others in the training set) and *imbalanced views* (some views are suited only for a subset of the classes). While the optimization problem given in Equation 5 deals with the imbalanced nature of the dataset, the multi-view aspect has not yet been considered.

Assuming that each view is more adapted only for a subset of classes, one possible way to implement this in a classifier is to associate a coefficient to each view based on its prediction. The better a view v recognizes a class c, the higher this coefficient $\beta_c^{(v)}$ should be. More precisely, the considered classifier is the following:

$$H(x) = \underset{c\in 1\dots K}{\operatorname{argmax}}\sum_{v=1}^{m}\beta_c^{(v)}\mathbb{I}[h^{(v)}(x)=c] \tag{7}$$

$$\text{where } \forall c\in\{1,\dots,K\}, \sum_{v=1}^{m}\beta_c^{(v)} = 1 \text{ and } \beta_c^{(v)}\in[0,1]. \tag{8}$$

In Equation 7, $h^{(v)}$ denotes the classifier learnt on view v and $\beta_c^{(v)}$ can be seen as the confidence that view v gets right for class c (hence the need for these coefficients to be positives). Due to the sum condition imposed in Equation 8, the coefficients $\beta_c^{(v)}$ also define how the views cooperate one with another, and we refer to them as *cooperation coefficients*.

In the following sections, we describe step by step how to learn a cooperation-based multi-view classifier as defined in Eq. 7, which is a solution to the optimization problem in Eq. 5 and subjected to the conditions of Eq. 8.

4 Multi-view Classification With Cooperation

The multi-view classifier in Eq. 7 associates to an example i the class k that obtains the highest score $\phi(i,k)$, computed as follows:

$$\phi(i,k) = \sum_{v \in \{1 \cdots m\}} \beta_k^{(v)} \mathbb{I}[h^{(v)}(i) = k]$$

If an example i is misclassified, then there exists at least one class $k \neq y_i$ such that $\phi(i,k) \geq \phi(i,y_i)$, that is, $e^{\phi(i,k)-\phi(i,y_i)} \geq 1$. On the other hand, i correctly classified implies that $\forall k \neq y_i, \phi(i,k) \leq \phi(i,y_i)$ and $0 < e^{\phi(i,k)-\phi(i,y_i)} \leq 1$. Basically, the exponential loss based on the score functions ϕ satisfies the conditions put on the losses in Eq. 6. Injecting these exponential losses in Equation 6, we have:

$$\begin{aligned} \|\mathbf{C}_{\mathcal{S}}\|_1 &\leq \sum_{i=1}^{n} \sum_{k \neq y_i} \frac{1}{n_{y_i}} \ell_{y_i,k}(h, x_i) \leq \sum_{i=1}^{n} \sum_{k \neq y_i} \frac{1}{n_{y_i}} e^{\phi(i,k)-\phi(i,y_i)} \\ &= \sum_{i=1}^{n} \sum_{k \neq y_i} \frac{1}{n_{y_i}} e^{\sum_{v=1}^{m} (\beta_k^{(v)} \mathbb{I}[h^{(v)}(i)=k] - \beta_{y_i}^{(v)} \mathbb{I}[h^{(v)}(i)=y_i])} \end{aligned} \tag{9}$$

A first approach to minimizing the norm of the confusion matrix is to find the classifiers and the cooperation coefficients that are the solution of the optimization problem given in Equation 9. This formulation is quite similar to the most general formulation of Multiple Kernel Learning (MKL) methods [2], where the kernels and the weighting coefficients are learnt at the same time. The differences are that in our case, the classifiers are not limited to kernels, the number of coefficients is higher and we use an exponential loss function.

The main advantage of this approach is that the cooperation between the views is promoted both in the training phase of the classifiers and in the choice of the coefficients. However, on the downside, the classifiers need to be learnt at the same time — which can make the learning procedure quite tedious — and it is not clear what the minimization goal for each classifier is. A friendlier (and easier to interpret) expression for Eq. 9 can be obtained by limiting the cooperation between the views only to the choice of the coefficients. The following result is a key step in this direction.

Lemma 1. *Let $n \in \mathbb{N}$ and $a_1, \ldots, a_n \leq 1$, so that $\sum_{j=1}^{n} a_j \leq 1$. Then the following inequality is true:*

$$\exp(\sum_{j=1}^{n} a_j) \leq \sum_{j=1}^{n} \exp(a_j). \tag{10}$$

Proof. The proof for this result is realized by induction : first we show that the result holds for two variables, then we prove the general case.

The first condition we set on the coefficients in the optimisation problem of Equation 8 is that their values should be in $[0, 1]$. The second condition requires that, for each class, the coefficients sum up to 1. Together, these conditions imply that the inner sums in Equation 9 take their values in $[-1, 1]$, same as each of their elements. We can thus apply lemma 1 to further simplify the expression given in Eq. 9.

$$\begin{aligned}
\|\mathbf{C}_{\mathcal{S}}\|_1 &\leq \sum_{i=1}^{n} \sum_{k \neq y_i} \frac{1}{n_{y_i}} e^{\sum_{v=1}^{m} (\beta_k^{(v)} \mathbb{I}[h_j(i)=k] - \beta_{y_i}^{(v)} \mathbb{I}[h^{(v)}(i)=y_i])} \\
&\leq \sum_{i=1}^{n} \sum_{k \neq y_i} \frac{1}{n_{y_i}} \sum_{v=1}^{m} e^{\beta_k^{(v)} \mathbb{I}[h^{(v)}(i)=k] - \beta_{y_i}^{(v)} \mathbb{I}[h^{(v)}(i)=y_i]} \\
&= \sum_{v=1}^{m} \sum_{i=1}^{n} \sum_{k \neq y_i} \frac{e^{\beta_k^{(v)} \mathbb{I}[h^{(v)}(i)=k] - \beta_{y_i}^{(v)} \mathbb{I}[h^{(v)}(i)=y_i]}}{n_{y_i}} \leq \sum_{v=1}^{m} \ell(h^{(v)}),
\end{aligned} \tag{11}$$

where $\ell(h^{(v)}) = \sum_{i=1}^{n} \sum_{k \neq y_i} \frac{1}{n_{y_i}} e^{\beta_k^{(v)} \mathbb{I}[h^{(v)}(i)=k] - \beta_{y_i}^{(v)} \mathbb{I}[h^{(v)}(i)=y_i]}$, defines the loss of the classifier $h^{(v)}$ — whose predictions are weighted by the coefficients $\beta_k^{(v)}, \forall k \in \{1, \ldots, K\}$ — on the training set S.

Minimizing the norm of the confusion matrix for a multi-view classifier H, as defined in Equation 7, ends up finding the classifiers and coefficients that minimize the loss $\sum_{v=1}^{m} \ell(h^{(v)})$. Remark that for fixed values of the coefficients, Equation 11 suggests that for each view, it suffices to find the classifier minimizing the loss $\ell(\cdot)$, instead of finding the classifiers that minimize a loss depending on their combination (which was the case in Equation 9).

Although the training procedure where all the coefficients and classifiers are learnt at the same time is still a viable solution, the previous remark suggests that a two-step procedure is better adapted for this case. The first step consists in finding, for each view, the classifier whose error $\ell(\cdot)$ is minimal. The second step consists in finding the coefficients that minimize the whole loss (Equation 11). Due to the formulation of Equation 11, the coefficients of one class can be computed independently from the coefficients of the other classes. More precisely, let $S_+^{(v)}$ denote the set of examples correctly classified by $h^{(v)}$ and $S_-^{(v)}$ the set

of misclassified ones. The right-handed side of Eq. 11 can be written as:

$$\sum_{v=1}^{m}\sum_{i=1}^{n}\sum_{k\neq y_i}\frac{1}{n_{y_i}}e^{\beta_k^{(v)}\mathbb{I}[h^{(v)}(i)=k]-\beta_{y_i}^{(v)}\mathbb{I}[h^{(v)}(i)=y_i]} =$$

$$\sum_{v=1}^{m}\sum_{i\in S_+^{(v)}}\frac{K-1}{n_{y_i}}e^{-\beta_{y_i}^{(v)}}+\sum_{v=1}^{m}\sum_{i\in S_-^{(v)}}\Big(\frac{K-2}{n_{y_i}}+\frac{e^{-\beta_{h^{(v)}(i)}^{(v)}}}{n_{y_i}}\Big) =$$

$$\sum_{c=1}^{K}\sum_{v=1}^{m}\Big(e^{-\beta_c^{(v)}}A+e^{\beta_c^{(v)}}B\Big)+C, \tag{12}$$

where:

$$A=\sum_{i\in S_+^{(v)}}\frac{K-1}{n_{y_i}}\mathbb{I}[y_i=c], B=\sum_{i\in S_-^{(v)}}\frac{1}{n_{y_i}}\mathbb{I}[h^{(v)}(i)=c]$$

$$\text{and } C=\sum_{c=1}^{K}\sum_{v=1}^{m}\sum_{i\in S_-^{(v)}}\frac{K-2}{n_{y_i}}\mathbb{I}[h^{(v)}(i)=c].$$

The first equality is obtained by splitting the training sample in $S_+^{(v)}$ and $S_-^{(v)}$, for all views v. In the third equality, we replace y_i and $h^{(v)}(i)$ by a label c, which allows us to regroup the non constant terms (that is, those depending on $\beta_c^{(v)}$).

Equation 12 suggests that for a given class c, the coefficients $\beta_c^{(v)}, \forall v \in \{1,\ldots,m\}$ should be the ones that minimise the *per class* loss $\ell(c)$:

$$\sum_{v=1}^{m}\Big(e^{-\beta_c^{(v)}}A+e^{\beta_c^{(v)}}B\Big). \tag{13}$$

The pseudo-code of the two-step method is given in Algorithm 1. The *stopping criterion* can be related to the empirical loss of the classifier computed on S, or the drop of the loss in Eq. 12, and so on. Contrary to the MKL-like approach suggested in Eq. 9, in the two-step method, the classifiers can be learnt in parallel, reducing the training time for the algorithm.

5 Boosting The Cooperation

5.1 A multi-view boosting method from the confusion matrix norm minimization

The two-step procedure in Algorithm 1 suggests that at each iteration the learnt classifiers should minimize some training error weighted by cooperation coefficients associated to the training examples. More precisely, the training procedure for each view consists in learning a classifier that minimizes a weighted empirical error, re-weighting the examples based on the performances of the classifier and

Algorithm 1 Carako

Given :
S = $\{(x_i, y_i)\}_{i=1}^n$ where $x_i \in \mathcal{X}^{(1)} \times \mathcal{X}^{(2)} \times ... \times \mathcal{X}^{(m)}$, $y_i \in \{1, ..., K\}$
Initialise :
For all $1 \leq c \leq K, 1 \leq v \leq m, \beta_c^{(v)} =$ rand(0,1) (w.r.t. to Eq. 8)

while *stopping criterion* not met:

1. $\forall v \in \{1, \ldots, m\}$, train $h^{(v)}$ minimizing the loss $\ell(h^{(v)})$ (eq. 11)
2. $\forall c \in \{1, \ldots, K\}$, compute $\beta_c^{(v)}$ minimizing the loss $\ell(c)$ (eq. 13), *w.r.t.* conditions in Eq. 8

Output :

$$H(\cdot) = \text{argmax}_{c \in \{1, \cdots, K\}} \sum_{v=1}^{m} \beta_c^{(v)} \mathbb{I}[h^{(v)}(\cdot) = c]$$

reiterating until a stopping criterion is met. Interestingly, this procedure is fairly similar to iterative boosting methods, such as AdaBoost [6] and its multi-class formulation AdaBoost.MM, recalled in Section 2.2. In this section, we study the case where the classifiers $h^{(v)}$ in Equation 11 are replaced with boosted classifiers.

An iterative boosting method runs for T rounds and its output hypothesis is computed as follows:

$$h(\cdot) = \text{argmax}_{c \in \{1, \ldots, K\}} \sum_{t=1}^{T} \alpha_t \mathbb{I}[h_t(\cdot) = c],$$

where, h_t are classifiers performing slightly better than random guessing (also called *weak classifiers*) and α_t are positive real-valued coefficients that represent the importance given to h_t. The main advantage of weak classifiers is that they can be used to exploit localized informations. Thus the motivation for replacing the per-view classifiers in Eq. 11 with multiple weak classifiers learnt on the views is to better use the localized information in each view, in particular information related to how the view recognizes the various classes.

When defining the multi-view classifier in Equation 7, we argued that each classifier should be associated with an importance coefficient depending on the prediction. However, having a single coefficient per class and view might not be a good strategy when dealing with boosted classifiers, since each of the weak classifiers has different performances. Thus we propose to associate to each classifier not only its importance coefficient, but also coefficients depending on its actual prediction:

$$h^{(v)}(\cdot) = \text{argmax}_{c \in \{1, \ldots, K\}} \sum_{t=1}^{T} \alpha_t^{(v)} \beta_{t,c}^{(v)} \mathbb{I}[h_t^{(v)}(\cdot) = c], \text{ where } \beta_{t,c}^{(v)} \in [0, 1]. \quad (14)$$

In this case the coefficient associated to a view v for a class c is :

$$\beta_c^{(v)} = (\sum_{t=1}^{T} \alpha_t^{(v)} \beta_{t,c}^{(v)}) / \sum_{t=1}^{T} \alpha_t^{(v)}.$$

The condition $\forall t \in \{1,\dots,T\}, \forall c \in \{1,\dots,K\}, \sum_{v=1}^{m} \beta_{t,c}^{(v)} = 1$ ensures that $\beta_c^{(v)}$ is always smaller than 1. The downside is that it does not guarantee that $\forall c \in \{1,\dots,K\}, \sum_{v=1}^{m} \beta_c^{(v)} = 1$, but rather that $\sum_{v=1}^{m} \beta_c^{(v)} \leq 1$.

The classifier defined in Equation 14 is similar to the classifier defined in Equation 7: for a given example, it computes a score for each class. In particular for examples in the training sample $S = \{(\mathbf{x}_i, y_i)\}_{i=1}^{n}$, these scores correspond to:

$$f_T^{(v)}(i,c) = \sum_{t=1}^{T} \alpha_t^{(v)} \beta_{t,c}^{(v)} \mathbb{I}[h_t^{(v)}(i) = c], \text{ for } 1 \leq v \leq m, 1 \leq c \leq K, 1 \leq i \leq n.$$

Armed with these score functions, we are now ready to tackle the last optimization problem in this paper. Continuing from Equation 11, we have:

$$\begin{aligned}
\|\mathbf{C}_{\mathcal{S}}\|_1 &\leq \sum_{v=1}^{m}\sum_{i=1}^{n}\sum_{c\neq y_i} \frac{1}{n_{y_i}} e^{\beta_c^{(v)}\mathbb{I}[h^{(v)}(i)=c] - \beta_{y_i}^{(v)}\mathbb{I}[h^{(v)}(i)=y_i]} \\
&\leq \sum_{v=1}^{m}\sum_{i=1}^{n}\sum_{c\neq y_i} \frac{1}{n_{y_i}} e^{\mathbb{I}[h^{(v)}(i)=c]} \leq \sum_{v=1}^{m}\sum_{i=1}^{n}\sum_{c\neq y_i} \frac{1}{n_{y_i}} e^{\ln(2+\exp\left[f_T^{(v)}(i,c) - f_T^{(v)}(i,y_i)\right])} \\
&= \sum_{v,i,c\neq y_i} \frac{1}{n_{y_i}} e^{\Delta_{f,T}(v,c,y_i)} + 2mK(K-1), \qquad (15)
\end{aligned}$$

where: $\Delta_{f,T}(v,c,y_i) = f_T^{(v)}(i,c) - f_T^{(v)}(i,y_i)$.

The second inequality follows from the fact that the coefficients $\beta_c^{(v)}$ are positive and at most 1. For the third inequality a particular case of the logistic loss is used. If $h^{(v)}$ predicts class c for example i ($\mathbb{I}[h^{(v)}(i) = c] = 1$), then $f_T^{(v)}(i,c) \geq f_T^{(v)}(i,y_i)$ and $\exp(f_T^{(v)}(i,c) - f_T^{(v)}(i,y_i)) \geq 1$. Since the difference between the scores may be infinitely small, we use 2 in the logistic loss, which ensures $\ln(2 + \exp\left[f_T^{(v)}(i,c) - f_T^{(v)}(i,y_i)\right]) \geq 1$. In the case where $h^{(v)}$ predicts another class for i, other than c, then $ln(2 + \exp\left[f_T^{(v)}(i,c) - f_T^{(v)}(i,y_i)\right]) > 0$.

We have thus:

$$\|\mathbf{C}_{\mathcal{S}}\|_1 \leq \sum_{v=1}^{m} \ell(h^{(v)}) + 2mK(K-1), \qquad (16)$$

where $\ell(h^{(v)}) = \sum_{i=1}^{n}\sum_{c\neq y_i} \frac{1}{n_{y_i}} \exp\Big(f_T^{(v)}(i,c) - f_T^{(v)}(i,y_i)\Big)$, defines the loss of the combinations of all the classifiers learnt on view v. Equation 16 suggests that at each iteration, for each view, a classifier that minimizes the loss $\ell(h_j)$ should be

learnt. It is interesting to notice that the loss $\ell(h^{(v)})$ is quite similar to the loss of AdaBoost.MM, as given in Equation 1 in Section 2, except for the coefficients $\beta_{c,t}^{(v)}$ and the re-weighting coefficients $\frac{1}{n_{y_i}}$. In other words, an AdaBoost.MM-like process should take place in each view. It follows that a cost matrix should be maintained for each view and the classifier (and its importance coefficient) should be computed in a similar way as in AdaBoost.MM or in MuMBo [13].

Algorithm 2 μCoMBo : MUlti-view COnfusion Matrix BOosting

Given

- $\mathrm{S} = \{(x_1, y_1), ..., (x_n, y_n)\}$ where $x_i \in \mathcal{X}^{(1)} \times ... \times \mathcal{X}^{(m)}$, $y_i \in \{1, ..., K\}$
- T the number of iterations
- $\forall i \in \{1 \dots n\}$, $\forall v \in \{1 \dots m\}$, $\forall c \in \{1 \dots K\}$

$$f_0^{(v)}(i, c) = 0, \beta_{0,c}^{(v)} = 1/m \text{ and}$$

$$\mathbf{D}_0^{(v)}(i, c) = \begin{cases} \frac{1}{n_{y_i}} & \text{if } c \neq y_i \\ -\frac{K-1}{n_{y_i}} & \text{if } c = y_i \end{cases}$$

for $t = 1$ **to** T **do**

$\forall v$: Get $h_t^{(v)}$ and $\alpha_t^{(v)} = \frac{1}{2} \ln \frac{1+\delta_t^{(v)}}{1-\delta_t^{(v)}}$, where $\delta_t = \frac{-\sum_{i=1}^n \mathbf{D}_{t-1}^{(v)}(i, h_t^{(v)}(\mathbf{x}_i))}{\sum_{i=1}^n \sum_{c \neq y_i} \mathbf{D}_{t-1}^{(v)}(i,c)}$

Compute $\beta_{t,c}^{(v)}$, $\forall v \in \{1 \dots m\}, c \in \{1 \dots K\}$ minimizing Eq. 15

Update cost matrices (for each $v = 1 \dots m$):

$$\mathbf{D}_t^{(v)}(i, c) = \begin{cases} \frac{1}{n_{y_i}} e^{f_t^{(v)}(i,c) - f_t^{(v)}(i,y_i)} & \text{if } c \neq y_i \\ -\sum\limits_{l \neq y_i}^{K} \frac{e^{f_t^{(v)}(i,l) - f_t^{(v)}(i,y_i)}}{n_{y_i}} & \text{if } c = y_i \end{cases}$$

where $f_t^{(v)}(i, c) = \sum\limits_{z=1}^{t} \alpha_z^{(v)} \beta_{z,c}^{(v)} \mathbb{I}1[h_z^{(v)}(i) = c]$

end for

Output final hypothesis :

$$H(\cdot) = \mathrm{argmax}_{c \in \{1,...,K\}} \sum_{t=1}^{T} \sum_{v=1}^{m} \beta_{t,c}^{(v)} \alpha_t^{(v)} \mathbb{I}[h_t^{(v)}(\cdot) = c]$$

Similarly to the two-steps procedures in Carako (Algorithm 1), the algorithm μCoMBo derived from the minimization of the loss in Equation 16 uses a two step procedure at each iteration: first a classifier is learnt for each view and the importance coefficient is computed as in Algorithm 2, and second the cooperation coefficients are computed so that they minimize the loss. The pseudo-code of μCoMBo is given in algorithm 2. Due to the boosting nature of the method, the stopping criterion used is the number of iterations.

5.2 On the theoretical properties of μCoMBo

One of the most important properties of (adaptive) boosting algorithms is that the total loss (Eq. 1 for AdaBoost.MM) drops at each iteration, provided that the chosen classifier weight verifies the boostability conditions. In the binary case, the classifier is required to perform slightly better than random guessing (that is, its error should be smaller than 0.5). For multi-class algorithms, such as AdaBoost.MM and μCoMBo, the condition consists in finding classifiers whose classification cost (that is, error) should be smaller that the one for an arbitrary baseline. Theorem 1 states that, for a given view, if the selected classifier verifies the multi-class boostability condition, then the total loss of μCoMBo (computed from that view) decreases.

Theorem 1. *For a given view $v \in \{1, \ldots, m\}$, suppose the cost matrix $\mathbf{D}_t^{(v)}$ is chosen as in the Algorithm 2, and the returned classifier $h_t^{(v)}$ satisfies the edge condition for the baseline $\mathbf{U}_{\delta_t^{(v)}}$ and cost matrix $\mathbf{D}_t^{(v)}$,* i.e. $\mathbf{D}_t^{(v)} \cdot \mathbf{1}_{h_t^{(v)}} \leq \mathbf{D}_t^{(v)} \cdot \mathbf{U}_{\delta_t^{(v)}}$, *where $\mathbf{1}_h$ is the matrix defined as $\mathbf{1}_h(i,l) = \mathbb{I}[h(i) = l]$.*

Then choosing a weight $\alpha_t^{(v)} > 0$ for $h_t^{(v)}$, allows

$$\ell_t(h^{(v)}) \leq \kappa_t^{(v)} \ell_{t-1}(h^{(v)}),$$

to hold, with:

$$\kappa_t^{(v)} = 1 - \frac{1}{2}\left(e^{\alpha_t^{(v)}} - e^{-\alpha_t^{(v)}}\right)\delta_t^{(v)} + \frac{1}{2}\left(e^{\alpha_t^{(v)}} + e^{-\alpha_t^{(v)}} - 2\right)$$

Proof. The proof is similar to the one provided for AdaBoost.MM in [16].

The result given in Theorem 1 implies that choosing the importance coefficient as in Algorithm 2, the drop of the loss for a view v is $\sqrt{1-\delta_t^{(v)}}$. That is, at each iteration, after the first step which consists in choosing a classifier per view, the total drop in loss is:

$$\sum_{v=1}^{m} \ell_t(h^{(v)}) \leq \sum_{v=1}^{m} \sqrt{1-\delta_t^{(v)}}\, \ell_{t-1}(h^{(v)}). \tag{17}$$

As long as the classifiers learnt on the views achieve positive edges on their corresponding cost matrices, the whole loss is guaranteed to decrease.

Note that in the right side of Equation 17, the loss $\ell_{t-1}(h^{(v)})$ depends on the score functions defined as:

$$f_t^{(v)}(i,c) = \sum_{z=1}^{t-1} \alpha_z^{(v)} \beta_{z,c}^{(v)} \mathbb{I}[h_z^{(v)}(i) = c] + \alpha_t^{(v)} \beta_{t,c}^{(v)} \mathbb{I}[h_t^{(v)}(i) = c],$$

since the classifiers are learnt before the cooperation coefficients β, that is, we simply suppose that all these coefficients are equal to 1. Obviously the drop in

the final loss, after the second step of iteration t, is much bigger given that all the β take their values in $[0, 1]$ and at most one of the $\beta_{t,c}^{(v)}$ is at 1 for a given class c. However, since these coefficients are computed as a solution to an optimization problem, finding an analytical expression for the actual drop after iteration t is quite challenging.

6 Experimental results

Experimental results of algorithms CARAKO and μCoMBo are compared with results obtained with state-of-the-art methods.

These experiments are intended to ascertain the relevance of the herein proposed algorithms, by checking the experimental gaps achieved when comparing mono and multi-view confusion imbalanced within a multi-class setting. As such, no tuning of hyperparameters was done (default hyper-parameters are considered), and only one – reduced – dataset was considered.

In order to have a first insight into the relevance of the approaches, we processed experiments on a subset of the Animal with Attributes (AwA) images open database proposed in [14], because it regroups the two problems addressed in this paper: (1) AwA is highly imbalanced: some classes are way more represented than others, and (2) AwA comes with six pre-extracted feature representations (thus, views) for each image, related to different properties of the images.

6.1 AwA presentation and experimental protocols

Originally, AwA comes with 50 classes and six views on images. We extracted[2] six classes, from the less represented to the most populated; class names (and number of examples) are: *beaver* (184), *buffalo* (559), *deer* (1072), *gorilla* (802), *lion* (483), and *polar+bear* (815). Four views (for a total of 6940 real attributes) among six were selected both on the nature of information contained therein (local versus global) and the possibility of the view to recognize all the classes or some of them. More precisely, we consider: Color Histogram features (2688 attributes), Local Self-Similarity features (2000 attributes), PyramidHOG (PHOG) (252 attributes), and Scale-Invariant Feature Transform features (2000 attributes).

Some examples of the animal classes are given in Figure 3.

Classifiers Six methods are tested: (1-2) early fusion: AdaBoost.MM and μCoMBo on the concatenation of the views, (3-4) late fusion: AdaBoost.MM and CARAKO, (5) multi-view methods: μCoMBo, and (6) multiclassMKL from the Shogun toolbox ([19]).

[2] The original dataset comes with a high number of examples (more than 30000), classes (50), and real features (10520). Currently, intensive experiments are on their way, with grid-search hyper-parameters tuning over more learning methods, running over the while AwA datasets: it should take several months.

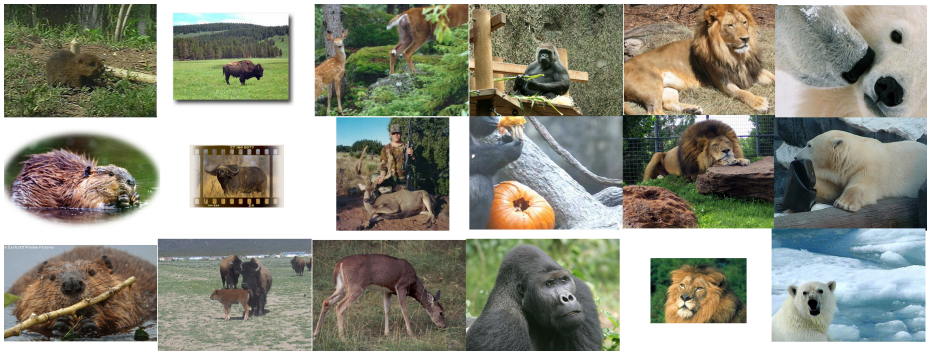


Fig. 3: Some examples of the six selected classes of AwA.

For the boosting-based methods, we used 1-level decision trees (stumps) as weak learners and they were run for 100 iterations; for Carako, the depth of the trees was limited to 10. The MKL were learnt with gaussian kernels on each view (same parameters: mean=8, width=2), with a regularized L1 norm and the regularization parameter $C = 1.$, and $\epsilon = 0.001$).

Evaluation protocol Through a 5-folds cross-validation process, we evaluated each classifier along five measures: recall per class, overall accuracy, MAUC, G-Mean, and norm of the confusion matrix. If K is the number of classes:

$$\text{MAUC} = \sum_{i \neq j} \frac{\text{AUC}_{i,j}}{K(K-1)} \text{ and G-mean} = \left(\prod_{j=1..K} \text{recall}_j \right)^{\frac{1}{K}}$$

G-mean and MAUC consider each class independently from its population in the learning sample; a G-mean is zero whenever one minor class has a zero recall.

6.2 Performance results

Table 1 gives the results for Accuracy, Gmean, MAUC and norm of the confusion matrix, as well as an indication of the training time[3]. About the overall accuracy, the better performing method is MultiClassMKL, while the recalls per class indicate that MKL focus on majority class, mechanically improving the overall accuracy. Second best method is μCoMBo, which is encouraging since it means that adding the cooperation between the views leads to good results while promoting equity among imbalanced classes. The difference between early CoMBo and μCoMBo relies on the fact that the latter encourages such a cooperation among views, while the former only learns a model after the normalized concatenation of all the views. It is then worth noticing that multi-view learning

[3] The MulticlassMKL is quite long for it is a QCQP problem, hardly depending on the number of classes and views (kernels); boosting approaches benefits from the possibility of parallelizing the weak classifiers training.

Table 1: Overall accuracy, per class recall, MAUC, G-mean and norm of the confusion matrix, for the six methods on AwA (the suffix `-e` indicates an early fusion). Results in boldface indicate when one algorithm is significantly better than the others (student test with 95% confidence). The rate of train examples is given under each class. The training time of a 10-folds cross-validation is given in minutes, on a 4-cores processor with limited RAM.

Algos	Acc.	*beaver*	*buffalo*	*deer*	*gorilla*	*lion*	*p.bear*	MAUC	Gmean	$\|\mathbf{C}\|$	time
		4.7%	14.3%	27.4%	20.5%	12.3%	20.8%				
Ada.MM-e	40.2%	0.0%	0.0%	77.8%	68.1%	0.0%	22.8%	0.694	0.000	1.385	90
Ada.MM-l	44.8%	0.0%	0.0%	**93.7%**	34.9%	0.0%	57.9%	0.731	0.000	1.594	71
Carako	34.4%	23.1%	20.1%	36.4%	33.5%	25.0%	50.7%	0.631	0.069	0.964	122
MKL	**58.6%**	0.0%	19.9%	77.3%	73.4%	13.6%	**85.6%**	0.669	0.000	0.980	2825
μCoMBo-e	43.4%	28.6%	30.8%	28.3%	66.9%	38.7%	55.6%	0.731	0.365	0.899	98
μCoMBo	55.5%	**44.5%**	**35.3%**	48.3%	**71.9** %	**38.8** %	74.4%	**0.821**	**0.481**	**0.552**	75

(μCoMBo) actually helps to achieve better results when minimizing a bound of the confusion norm when facing class-imbalanced datasets.

Concerning the per-class recalls, early-μCoMBo, Carako, and the multi-view μCoMBo all tend to reduce the impact of majority classes, while focusing on the minority ones. This behavior was expected through the minimization of the confusion norm. Symmetrically, AdaBoost.MM (both early and late fusion) and MKL clearly all favor the majority classes over the minority ones.

About measures dedicated to multi-class approaches, G-mean and MAUC point out the smoothing effect of μCoMBo on errors which helps to better take into consideration the minority classes. According to G-mean and MAUC, the results strongly suggest that the best method is μCoMBo, which was expected since μCoMBo was designed to deal with imbalanced multi-view datasets. As for accuracy, the performances of the various methods on the G-mean imply that both μCoMBo and Carako promote some sort of leveling process among the classes, thus a better equity among them; however, μCoMBo ends up with better results than Carako, thanks to the cooperation among views. AdaBoost.MM and MKL have poor G-mean since they fail to recognize *beaver*, the minority class.

7 Discussion

This work merges imbalanced multi-view and multi-class learning, and proposes a boosting-like algorithm to address it. As far as we know, albeit many even recent results about ensemble-based imbalanced multi-class learning have been published [7, 22, 5, 9, 3, 23], no other approach has emerged that would meanwhile consider the capabilities of multi-view diversity of information sources. As a consequence, the proposed approach here is quite original, and cannot be fairly compared with any other state-of-the-art theory or algorithm.

Our preliminary experimental results show that our approach seems to be relevant for facing imbalanced views and classes, together with theoretical guarantees. As such, this work raises several questions and prospect works.

On the confusion matrix In Equations 4, 5 and 6, the operator norm of the confusion matrix is bounded by the $l1$-norm. First, remark that due to the equivalence between norms in finite dimension, minimizing the (entry-wise) $l1$-norm is a viable alternative to the original goal and it ensures that the learning procedure outputs a classifier with a low error. Second, as briefly commented upon in Section 3, it is quite difficult to find an analytical expression for the operator norm of the confusion matrix in the supervised setting as given in Definition 1. In order to bypass these shortcomings and to tackle the original goal (minimizing the operator norm of the confusion matrix), future works will be focused on exploring alternative definitions for the confusion matrix, such as loss-based confusion matrices as in [17], entry-wise decomposition of the confusion matrix as in [15], three-dimensional tensors, etc.

On the optimized norm Aside from the choice of the confusion matrix, another research question touched upon in this paper is the choice of the norm. We think that it would be interesting to define and study other norms, such as the $l1$-norm, other p-norms, or even more exotic norms. In particular, a challenging problem is the definition of confusion matrices and confusion matrices' norms for the *multi-view setting* either as a generalization of the usual definitions to the three-dimensional tensor, or based on the tensor's theory.

On the loss functions The main advantage of the result in Equation 6 is the flexibility of the loss functions. Although our work is mainly based on the exponential losses, Equations 9 and 15 show that other information can be embedded in the losses. As such, Equation 6 can be used to derive other (novel) multi-view imbalanced learning methods by either choosing other loss functions, or modifying the information contained therein (such as enforcing the cooperation between the views, embedding prior information on the classes, etc.).

On the combined learners In Section 4, we argued that the main advantage of Carako (Algorithm 1) over MKL is the fact that our method is not limited to kernel methods. Other, more empirical, works will be focused on testing Carako with other learning methods and studying the effect that the cooperation between the views has on the final combined classifier.

On theoretical improvements Finally, future work will also be focused on finding tighter bounds for the result given in Equation 15. As is, the constant term (right-handed side of the equation) depends on the number of classes and when this number is important, the constant might overshadow the true objective in the left-handed side of the equation. Although this might not present a real challenge for current multi-view imbalanced datasets, we think that finding tighter bounds will not only address a crucial issue for our approach, but it might also allow to derive novel algorithms in the same spirit as μCoMBo.

8 Conclusion

In this paper, we proposed various multi-view ensemble learning methods, proposed in Sections 4 and 5, for dealing with imbalanced views and classes. The novelty of our approach consists in injecting a cooperation-based multi-view classifier (Eq. 7) in the imbalanced classes framework (Eq. 5). This choice is mainly motivated by the promotion of the cooperation between the views in the output space, so that each view is associated with the classes it recognizes best. Our intuition is further confirmed by the empirical results in Section 6. We think that the work presented here is a first clear answer to the question posed in the introduction, while at the same time raising various research questions (e.g. the choices of the confusion matrix, its norm, the multi-view classifier, etc.). In the next future, a deep study of the complexity of μCoMBo is required, which mainly involves the specific properties of the non-linear convex optimization it relies on.

References

1. Ayache, S., Quénot, G., Gensel, J.: Classifier fusion for svm-based multimedia semantic indexing. In: ECIR. pp. 494–504 (2007)
2. Bach, F.R., Lanckriet, G.R.G.: Multiple kernel learning, conic duality, and the smo algorithm. In: In Proceedings of the 21st International Conference on Machine Learning (ICML (2004)
3. Bi, J., Zhang, C.: An empirical comparison on state-of-the-art multi-class imbalance learning algorithms and a new diversified ensemble learning scheme. Knowledge-Based Systems (2018)
4. Chawla, N.V., Bowyer, K.W., Hall, L.O., Kegelmeyer, W.P.: Smote: synthetic minority over-sampling technique. J. Artif. Int. Res. **16**(1), 321–357 (Jun 2002)
5. Fernández, A., Garcia, S., Herrera, F., Chawla, N.V.: Smote for learning from imbalanced data: Progress and challenges, marking the 15-year anniversary. Journal of Artificial Intelligence Research **61**, 863–905 (2018)
6. Freund, Y., Schapire, R.E.: Experiments with a new boosting algorithm. In: In Proceedings of the International Conference on Machine Learning. pp. 148–156 (1996)
7. Garca, S., Zhang, Z.L., Altalhi, A., Alshomrani, S., Herrera, F.: Dynamic ensemble selection for multi-class imbalanced datasets. Information Sciences **445-446**, 22 – 37 (2018)
8. Gehler, P., Nowozin, S.: On feature combination for multiclass object classification. In: Computer Vision, 2009 IEEE 12th International Conference on. pp. 221–228 (2009)
9. Haixiang, G., Yijing, L., Shang, J., Mingyun, G., Yuanyue, H., Bing, G.: Learning from class-imbalanced data: Review of methods and applications. Expert Systems with Applications **73**, 220 – 239 (2017)
10. He, H., Garcia, E.A.: Learning from imbalanced data. IEEE Trans. on Knowl. and Data Eng. **21**(9), 1263–1284 (Sep 2009)
11. Huusari, R., Kadri, H., Capponi, C.: Multiview metric learning in vector-valued kernel spaces. In: Aistats, 2018. Lanzarote, Spain (2018)

12. Kadri, H., Ayache, S., Capponi, C., Koço, S., Dupé, F.X., Morvant, E.: The multi-task learning view of multimodal data. In: Asian Conference on Machine Learning, JMLR. pp. 261–276 (2013)
13. Koço, S., Capponi, C.: A boosting approach to multiview classification with co-operation. In: European Conference on Machine Learning (ECML). vol. 6912, pp. 209–228 (2011)
14. Lampert, C.H., Nickisch, H., Harmeling, S.: Learning to detect unseen object classes by between-class attribute transfer. In: Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on. pp. 951–958 (2009)
15. Morvant, E., Koço, S., Ralaivola, L.: PAC-Bayesian Generalization Bound on Confusion Matrix for Multi-Class Classification. In: International Conference on Machine Learning. pp. 815–822 (2012)
16. Mukherjee, I., Schapire, R.E.: A theory of multiclass boosting. J. Mach. Learn. Res. **14**(1), 437–497 (Feb 2013)
17. Ralaivola, L.: Confusion-based online learning and a passive-aggressive scheme. In: Neural Information Processing Systems Conference (2012)
18. Snoek, C., Worring, M., Smeulders, A.: Early versus late fusion in semantic video analysis. In: Proceedings of the 13th annual ACM international conference on Multimedia. pp. 399–402. MULTIMEDIA '05, ACM, New York, NY, USA (2005)
19. Sonnenburg, S., Raetsch, G., Henschel, S., Widmer, C., Behr, J., Zien, A., de Bona, F., Binder, A., Gehl, C., Franc, V.: The shogun machine learning toolbox. Journal of Machine Learning Research **11**, 1799–1802 (june 2010)
20. Sun, Y., Kamel, M., Wang, Y.: Boosting for learning multiple classes with imbalanced class distribution. In: In 2006 IEEE International Conference on Data Mining, HongKong. pp. 592–602 (2006)
21. Wang, S., Yao, X.: Multiclass imbalance problems: Analysis and potential solutions. Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on **42**(4), 1119–1130 (2012)
22. Wu, F., Jing, X.Y., Shan, S., Zuo, W., Yang, J.Y.: Multiset feature learning for highly imbalanced data classification. In: AAAI. pp. 1583–1589 (2017)
23. Yijing, L., Haixiang, G., Xiao, L., Yanan, L., Jinling, L.: Adapted ensemble classification algorithm based on multiple classifier system and feature selection for classifying multi-class imbalanced data. Knowledge-Based Systems **94**, 88 – 104 (2016)