Exploring hierarchical machine learning for hardware-limited multi-class inference on compressed measurements

Wissam Benjilali, William Guicquero, Laurent Jacques, Gilles Sicard

Exploring Hierarchical Machine Learning for Hardware-Limited Multi-Class Inference on Compressed Measurements

Wissam Benjilali[1], William Guicquero[1], Laurent Jacques[2] and Gilles Sicard[1]
[1]Univ. Grenoble Alpes, CEA, LETI, F-38000 Grenoble, France
[2]ISPGroup, ICTEAM/ELEN, UCLouvain, Louvain-la-Neuve, Belgium
Email: {wissam.benjilali, william.guicquero, gilles.sicard}@cea.fr; laurent.jacques@uclouvain.be

***Abstract*—This paper explores hierarchical clustering methods to learn a hierarchical multi-class classifier on compressed measurements in the context of highly constrained hardware (*e.g.*, always-on ultra low power vision systems). In contrast to the popular multi-class classification approaches based on multiple binary classifiers (*i.e.*, one-vs.-all and one-vs.one [1]), a hierarchical classifier requires only $\mathcal{O}(\log_2 C)$ binary classifiers in a decision tree. In this work, we investigate three clustering methods used to construct balanced clusters at each node thus reducing the depth of the decision tree in order to lower hardware requirements to its minimum. A binary Support Vector Machine (SVM) [2] classifier is then learned on Compressive Sensing measurements [3] at each node of the hierarchical tree. Our results, based on two object recognition databases (AT&T and COIL-100 databases), show the competitiveness of hierarchical classification in terms of hardware requirements (lower memory and computational complexity) as well as its classification accuracy.**

***Index Terms*—Hierarchical learning, compressive sensing, clustering, Support Vector Machine, embedded multi-class inference**

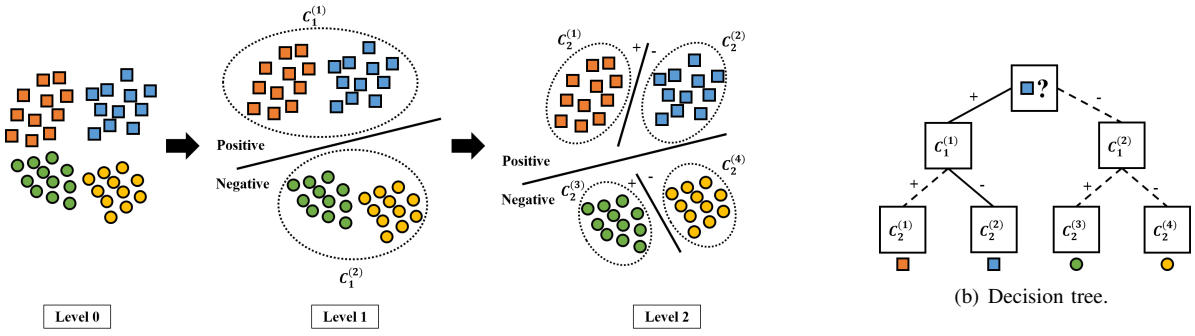
I. Introduction

Over the last decade, the trend in smart embedded systems consists in developing computational-friendly, always-on decision making systems. To achieve this goal, the design of new smart systems tends to take advantage of recent advances in signal acquisition schemes and inference algorithms, well optimized for low-power systems. Moreover, the design of this kind of information-retrieval signal processing architectures has to deal with on-chip constraints related to the data dimensionality and algorithms complexity.

Recently, several alternative sensing schemes based on the Compressive Sensing theory (CS, [3]), [4]–[7] and dedicated embedded processing [8]–[10] have been investigated to relax on-chip constraints. Indeed, CS has emerged as a hardware-friendly data acquisition scheme that can dramatically reduce data dimensionality and thus sensor design constraints. On the other hand, hierarchical machine learning algorithms [11]–[14] can significantly reduce memory and computational requirements related to an embedded decision making algorithm. Considering a multi-class image classification system based on binary classifiers such as SVMs, two popular strategies are usually adopted [1]. The first one is called a one-vs.-all and involves the training of C classifiers for a C classes problem. The second one is the one-vs.-one strategy that trains $C(C-1)/2$ classifiers for the same C classes problem. However, thanks to its intrinsic nature, a hierarchical inference dynamically requires to run only $\mathcal{O}(\log_2 C)$ cascaded binary classifiers. Thus, for a highly constrained embedded system (*e.g.*, smart always-on sensor) it seems relevant to investigate hierarchical strategies on CS measurements to relax hardware requirements related to signal acquisition (*e.g.*, A/D conversion, power consumption) and data processing (*e.g.*, memory needs) to perform embedded multi-class inference.

Contributions: In our paper, we focus on hierarchical learning in the context of highly constrained hardware in order to reduce hardware requirements related to an embedded multi-class inference. We introduce new methods to construct the hierarchical tree to train a hierarchical classifier (binary decision tree) minimizing as a consequence the number of decision nodes, end thus, the number of SVM-based affine transform to perform at the inference. Using classes centroids and sample labels of a training database, three methods have been investigated to create two clusters at each node that are balanced in terms of number of classes: (Method 1): sequential K-means clustering, (Method 2): SVM-based clustering and (Method 3): a clustering based on the Principal Component Analysis (PCA). The proposed methods assumes specific priors on intra-class & inter-class data distribution to construct the decision tree. In Method 1 we propose a K-means-inspired [15] algorithm to construct two balanced clusters limiting the decision tree depth. In Method 2, we construct balanced clusters that directly maximize the soft margin of a SVM performed on samples data belonging to each class-clusters. Finally, Method 3 takes advantage of a new basis estimated from a PCA [16] that better represents the classes centroids variability to define a separation threshold identifying two classes clusters. In the following, we will present our proposed clustering methods as well as general considerations on hardware and an evaluation of classification accuracy for a basic inference problem. In the context of limited processing and memory resources, we consider CS measurements as raw data for both training and testing.

Laurent Jacques is funded by the Belgian F.R.S.-FNRS and by the project AlterSense (MIS-FNRS).



(a) Hierarchical learning.

(b) Decision tree.

Fig. 1: An illustration of the hierarchical learning (1(a)) and the inference decision tree (1(b)). In (a) The input multi-class dataset to be classified is presented at Level 0. A first balanced clustering (2 clusters, each associated to the same number of classes) is performed at Level 1, then a binary classifier is trained. This process is repeated for each cluster until the construction of a single-class cluster at each terminal node. Here, $C_i^{(j)}$ represents j^{th} cluster at level i. (b) shows the inference process in the case of a binary hierarchical learning given a test sample (represented by the blue square in this figure).

II. Hierarchical classification on CS: key concepts

Hierarchical learning on CS measurements refers to constructing the hierarchical decision tree directly in the CS domain. Let us consider the sensing matrix $\boldsymbol{A} \in \mathbb{R}^{M\times N}$ composed of $M \ll N$ measurement vectors that are properly designed to perform CS. It allows, for a N-length vector $\boldsymbol{x} \in \mathbb{R}^N$, to acquire a CS measurement vector using the sensing model described as $\tilde{\boldsymbol{x}} = \boldsymbol{A}\boldsymbol{x} \in \mathbb{R}^M$. In particular, CS matrices has to satisfy the Restricted Isometry Property (RIP) [17] with high probability. From a decision making point of view, it was shown that the RIP preserves the Euclidean distance in the CS domain [18] of low complexity signals (*e.g.*, k-sparse). When dealing with linearly separable convex sets, the rare eclipse problem [19] [20] provides a lower bound based on the distance between classes to preserve the disjointness in the CS domain.

III. Proposed hierarchical learning methods

A. Notations and background

Let us consider a database of N-length "vectors" in $\mathbb{R}^N$ (*e.g.*, signals with N samples, or images with N pixels) composed of C classes. This database is separated into two databases: a "train" set $\boldsymbol{X} \in \mathbb{R}^{N\times n_1 C}$, where each class is composed of n_1 samples, associated with labels $\boldsymbol{l} \in \{1,\cdots,C\}^{n_1 C}$; and a "test" set $\boldsymbol{Y} \in \mathbb{R}^{N\times n_2 C}$ with unknown labels and composed of n_2 samples per class. We refer to $\boldsymbol{X}^j = (\boldsymbol{X}_1^j,\cdots,\boldsymbol{X}_{n_1}^j) \in \mathbb{R}^{N\times n_1}$ and $\boldsymbol{Y}^j = (\boldsymbol{Y}_1^j,\cdots,\boldsymbol{Y}_{n_2}^j) \in \mathbb{R}^{N\times n_2}$ for the train and the test sets restricted to the j^{th} class, respectively. The notation $\boldsymbol{x} \in \boldsymbol{X}$ or $\boldsymbol{x} \in \boldsymbol{X}^j$, means that the sample $\boldsymbol{x}$ is an arbitrary column of $\boldsymbol{X}$ or $\boldsymbol{X}^j$, respectively (and similarly for $\boldsymbol{Y}$). The mean vectors of each class (i.e., class centroids) are expressed as $\boldsymbol{\mu}_j = \frac{1}{n_1}\sum_{i=1}^{n_1}\boldsymbol{X}_i^j$, for $1 \le j \le C$. The distance matrix containing the euclidean distances between the mean vectors will be denoted $\boldsymbol{D} = (\|\boldsymbol{\mu}_p - \boldsymbol{\mu}_q\|_2)_{1\le(p,q)\le C}$.

On the other hand, given $\{(\boldsymbol{x}_1,l_1),\ldots,(\boldsymbol{x}_k,l_k),\ldots,(\boldsymbol{x}_{2n_1},l_{2n_1})\} \subset \mathbb{R}^N\times\{-1,1\}$ samples of two different classes in $\boldsymbol{X}$. The binary SVM optimization problem between this two classes is written as:

$$\{\hat{\boldsymbol{\omega}},\hat{b},\hat{\boldsymbol{\xi}}\} = \underset{\boldsymbol{\omega}\in\mathbb{R}^N,b,\boldsymbol{\xi}\in\mathbb{R}^{2n_1}}{\arg\min} \left(\tfrac{1}{2}\|\boldsymbol{\omega}\|_2^2 + \lambda\sum_{k=1}^{2n_1}\xi_k\right)$$
$$\text{s.t.} \quad l_k(\boldsymbol{\omega}^\top\boldsymbol{x}_k + b) \ge 1-\xi_k,\ \xi_k \ge 0, 1\le k\le 2n_1, \quad (1)$$

where $\hat{\boldsymbol{\omega}}$ is the weight vector, $\hat{b}$ is the bias scalar, $\boldsymbol{\xi}$ is the vertical concatenation of $2n_2$ slack variables and λ is a regularization parameter. Once the binary classifier constructed, the canonical SVM inference can be expressed as an affine transformation. Thus, for a test sample $\boldsymbol{y} \in \boldsymbol{Y}$ the inferred class c_y is given by:

$$c_y = \text{sign}(\boldsymbol{\omega}^\top\boldsymbol{y} + b) \quad (2)$$

Note that in the context of an embedded classification system based on supervised learning, we generally consider two stages: i) learning the classifier parameters on a training set, off-line, ii) performing embedded in-line inference on sensed data. In this section, we will first present the proposed algorithms to construct a hierarchical tree, the inference algorithm and finally a discussion on hardware requirements for a basic inference application on original and compressed data.

B. Training the hierarchical classifier

As depicted in Fig. 1(a), the main idea of a hierarchical learning is to divide a set of classes into two subsets at every hierarchical node. In the rest of this subsection we will first present our proposed methods to create two balanced clusters in a hierarchical node given a set of classes (Algorithm 1-3). Secondly, we describe the algorithm used to create the decision tree (Fig. 1(b)), then used for the inference (Algorithm 4).

1) Sequential K-means clustering (Method 1): Given centroids of a set of classes and their corresponding distance matrix, Method 1 aims at creating two balanced clusters based on classes centroids. Inspired by a K-means, the algorithm of Method 1 is first initialized to the classes centroids maximizing the Euclidean constrained distances (*i.e.*, $\boldsymbol{c}_1$ and $\boldsymbol{c}_2$). Then, at each iteration we sequentially assign to each cluster the center minimizing the Euclidean distance to their centroids.

In addition, $\boldsymbol{c}_1$ and $\boldsymbol{c}_2$ are updated at each iteration, *i.e.*, we calculate the mean of their clusters, respectively. This process is repeated until the assignment of all initial centroids. Finally, the algorithm return two clusters $\mathcal{C}_1$ and $\mathcal{C}_2$ (see Algorithm 1). allowing to cluster a set of centroids at a given node.

Algorithm 1 Sequential K-means clustering (Method 1)

Input $\boldsymbol{\mu} \in \mathbb{R}^{N\times K}$ centroids of K classes in $\boldsymbol{X}$ and $\boldsymbol{D}$
$\{m_1, m_2\} \leftarrow \arg\max_{i,j} \boldsymbol{D}_{ij}$;
$\boldsymbol{c}_1 \leftarrow \boldsymbol{\mu}_{m_1}$; $\boldsymbol{c}_2 \leftarrow \boldsymbol{\mu}_{m_2}$;
$\mathcal{C}_1 \leftarrow \{\boldsymbol{c}_1\}$; $\mathcal{C}_2 \leftarrow \{\boldsymbol{c}_2\}$;
$\boldsymbol{\mu} \leftarrow \boldsymbol{\mu} \setminus \{\boldsymbol{\mu}_{m_1}, \boldsymbol{\mu}_{m_2}\}$;
while $\boldsymbol{\mu} \neq \{\emptyset\}$ **do**
  $m_1 \leftarrow \arg\min_j \left\|\boldsymbol{c}_1 - \boldsymbol{\mu}_j\right\|_2$;
  $m_2 \leftarrow \arg\min_j \left\|\boldsymbol{c}_2 - \boldsymbol{\mu}_j\right\|_2$;
  $\boldsymbol{c}_1 \leftarrow \boldsymbol{\mu}_{m_1}$; $\boldsymbol{c}_2 \leftarrow \boldsymbol{\mu}_{m_2}$;
  $\mathcal{C}_1 \leftarrow \mathcal{C}_1 \cup \{\boldsymbol{c}_1\}$; $\mathcal{C}_2 \leftarrow \mathcal{C}_2 \cup \{\boldsymbol{c}_2\}$;
  $\boldsymbol{\mu} \leftarrow \boldsymbol{\mu} \setminus \{\boldsymbol{\mu}_{m_1}, \boldsymbol{\mu}_{m_2}\}$;
  $\boldsymbol{c}_1 \leftarrow$ centroid of $\mathcal{C}_1$; $\boldsymbol{c}_2 \leftarrow$ centroid of $\mathcal{C}_1$;
end while
return $\mathcal{C}_1$ and $\mathcal{C}_2$

2) SVM based balanced clustering (Method 2): In this second method described in Algorithm 2, the algorithm is also initialized to the centroids maximizing the Euclidean centroid distance (*i.e.*, $\boldsymbol{c}_1$ and $\boldsymbol{c}_2$). At each iteration, the SVM (1) is trained on the samples associated to the current cluster centroids, then it assigns the class centroid $\boldsymbol{\mu}_{m_1}$ maximizing the positive margin to the first cluster centroid $\boldsymbol{c}_1$ and the class centroid $\boldsymbol{\mu}_{m_2}$ minimizing the negative margin to $\boldsymbol{c}_2$ using the affine function presented in (2). An update of each cluster labeled samples is finally performed at each iteration for the next SVM training. This process is then repeated until the assignment of all centroids (with related input class samples).

3) PCA based balanced clustering (Method 3): In the third method (see Algorithm 3), the clustering is performed thanks to an orthogonal projection in the PCA [16] domain. The main idea is to find a new basis that provide a better description of the data variability to divide the space into two balanced half-spaces. In the case of the PCA, the projection is defined as the first principal component of the initial centroids to avoid under-fitting due to classes distribution. Thus, once the projection is learned one can construct two clusters such that the centroids bellow the median of the projection in the PCA domain are assigned to the first cluster and the ones above this threshold are assigned to the second. This way, the median enables a balanced clustering.

Finally, the binary decision tree is recursively constructed using one of the balanced clustering proposed methods, training a binary SVM (*i.e.*, (1)) at each node. Algorithm 4, describes the decision tree construction overall algorithm that is then used for the inference.

Algorithm 2 SVM based clustering (Method 2)

Input K classes in $\boldsymbol{X}$, centroids $\boldsymbol{\mu} \in \mathbb{R}^{N\times K}$ and $\boldsymbol{D}$
$\{m_1, m_2\} \leftarrow \arg\max_{i,j} \boldsymbol{D}_{ij}$;
$\boldsymbol{c}_1 \leftarrow \boldsymbol{\mu}_{m_1}$; $\boldsymbol{c}_2 \leftarrow \boldsymbol{\mu}_{m_2}$;
$\mathcal{C}_1 \leftarrow \{\boldsymbol{X}^{m_1}\}$; $\mathcal{C}_2 \leftarrow \{\boldsymbol{X}^{m_2}\}$;
$\boldsymbol{\mu} = \boldsymbol{\mu} \setminus \{\boldsymbol{\mu}_{m_1}, \boldsymbol{\mu}_{m_2}\}$;
while $\boldsymbol{\mu} \neq \{\emptyset\}$ **do**
  associate $\mathcal{C}_1$ to $\{1\}^{n_1\text{card}(\mathcal{C}_1)}$;
  associate $\mathcal{C}_2$ to $\{-1\}^{n_1\text{card}(\mathcal{C}_2)}$;
  for all $\boldsymbol{x}_k \in \mathcal{C}_1 \cup \mathcal{C}_2$:
  $\{\hat{\boldsymbol{\omega}}, \hat{b}, \hat{\boldsymbol{\xi}}\} = \arg\min_{\boldsymbol{\omega},b,\boldsymbol{\xi}} \left(\frac{1}{2}\left\|\boldsymbol{\omega}\right\|_2^2 + \lambda \sum_k \xi_k\right)$
  s.t. $l_k(\boldsymbol{\omega}^\top \boldsymbol{x}_k + b) \geq 1 - \xi_k,\ \xi_k \geq 0,$
  $1 \leq k \leq n_1\text{card}(\mathcal{C}_1) + n_1\text{card}(\mathcal{C}_2)$.
  $m_1 = \arg\max_j \hat{\boldsymbol{\omega}}^\top \boldsymbol{\mu}_j + \hat{b}$;
  $m_2 = \arg\min_j \hat{\boldsymbol{\omega}}^\top \boldsymbol{\mu}_j + \hat{b}$;
  $\boldsymbol{c}_1 \leftarrow \boldsymbol{\mu}_{m_1}$; $\boldsymbol{c}_2 \leftarrow \boldsymbol{\mu}_{m_2}$;
  $\mathcal{C}_1 \leftarrow \mathcal{C}_1 \cup \{\boldsymbol{X}^{m_1}\}$; $\mathcal{C}_2 \leftarrow \mathcal{C}_2 \cup \{\boldsymbol{X}^{m_2}\}$;
  $\boldsymbol{\mu} = \boldsymbol{\mu} \setminus \{\boldsymbol{\mu}_{m_1}, \boldsymbol{\mu}_{m_2}\}$;
end while
return $\mathcal{C}_1$ and $\mathcal{C}_2$

Algorithm 3 PCA based clustering (Method 3)

Input $\boldsymbol{\mu} \in \mathbb{R}^{N\times K}$ centroids of K classes in $\boldsymbol{X}$
SVD [21] : $\boldsymbol{\mu} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\top$ s.t. $\boldsymbol{U} = [\boldsymbol{u}_1 \dots \boldsymbol{u}_{C-1}]$.
$\theta \leftarrow$ median value of $\boldsymbol{u_1}^\top \boldsymbol{\mu}$;
$\mathcal{C}_1 \leftarrow (\boldsymbol{\mu})_j \quad \text{s.t.} \quad \boldsymbol{u_1}^\top \boldsymbol{\mu}_j < \theta$;
$\mathcal{C}_2 \leftarrow (\boldsymbol{\mu})_j \quad \text{s.t.} \quad \boldsymbol{u_1}^\top \boldsymbol{\mu}_j \geq \theta$;
return $\mathcal{C}_1$ and $\mathcal{C}_2$

C. Testing the hierarchical-based inference

As previously mentioned, training the hierarchical tree allows to construct a binary decision tree where each path from a root to a leaf is associated to a decision rule defined by the binary test (2) learned by a SVM. For a new test sample $\boldsymbol{y} \in \boldsymbol{Y}$, a decision rule (2) is applied at every node where the margin sign is used to decide to which next branch the sample belongs to. Thus, the predicted class is provided by the path indicated by the successive decisions (see Fig. 1(b)).

D. Embedded resources requirements analysis

Table II summarizes embedded resources requirements related to a one-vs.-all approach, a one-vs.-one approach and the proposed binary hierarchical learning approach for a N-dimensional C-classes classification problem. In the context of an embedded system, since the training is performed in an off-line system, we are mainly interested into the requirements related to the inference part, *i.e.*, memory needs to store ex-situ learned patterns and computational complexity related to the inference. In fact, in the case of the one-vs.-all, C classifiers are learned and thus have to be stored to perform C N-dimensional projections [2]. For a one-vs.-one, $C(C-1)/2$ classifiers are learned. However, when using a hierarchical approach, the number of classifiers to learn is reduced to $C-1$

		Linear SVM		Hierarchical methods (SVM-based tree decision)			
		One-vs.-one	One-vs.-all	K-means	**Our methods**		
					Method 1	Method 2	Method 3
Accuracy (%)	AT&T (w/o CS)	96.15 ± 1.41	$\mathbf{96.75 \pm 1.56}$	90.07 ± 2.51	89.35 ± 3.30	$\mathbf{92.87 \pm 2.44}$	91.51 ± 2.27
	COIL (w/o CS)	$\mathbf{98.32 \pm 1.22}$	95.76 ± 1.42	$\mathbf{96.40 \pm 1.65}$	93.06 ± 2.10	$\mathbf{95.10 \pm 1.62}$	95.07 ± 1.25
	AT&T (w/ CS*)	$\mathbf{95.52 \pm 1.61}$	95.03 ± 1.56	87.80 ± 2.35	84.75 ± 3.91	$\mathbf{90.41 \pm 2.44}$	89.85 ± 2.50
	COIL (w/ CS*)	$\mathbf{97.82 \pm 1.02}$	93.71 ± 1.32	$\mathbf{94.17 \pm 1.57}$	87.30 ± 3.03	$\mathbf{92.09 \pm 1.83}$	91.82 ± 1.41
Nb. of SVMs**	AT&T	40	780	8	**6**	**6**	**6**
	COIL	32	496	7	**5**	**5**	**5**

TABLE I: Classification accuracy of our methods compared to one-vs.-all approach, one-vs.-one approach and the K-means based hierarchical learning. *with a Compression Ratio of 25%. **Number of projections to perform at the inference stage.

Algorithm 4 Decision tree construction

Input training set $\boldsymbol{X}$
$level \leftarrow 1$ and $node \leftarrow 1$
create two balanced clusters $\mathcal{C}_1^{(1)}$; $\mathcal{C}_1^{(2)}$ on $\boldsymbol{X}$;
solve (1) on $\mathcal{C}_1^{(1)}$ and $\mathcal{C}_1^{(2)}$ to learn $SVM_1^{(1)}$ classifier
while level $\neq$ ceil($\log_2 C$) $- 1$ **do**
  $level \leftarrow level + 1$ and $node \leftarrow 2^{level-1}$
  for n `in 1 to` $node$ **do**
    if card($\mathcal{C}_{level}^{(n)}$) $\neq n_1$ **then**
      Create two balanced clusters $\mathcal{C}_{level+1}^{(2n-1)}$ and $\mathcal{C}_{level+1}^{(2n)}$ on $\mathcal{C}_{level}^{(n)}$
      solve (1) on $\mathcal{C}_{level+1}^{(2n-1)}$ and $\mathcal{C}_{level+1}^{(2n)}$ to learn $SVM_{level+1}^{(n)}$ classifier
    end if
  end for
end while

to perform only $\lceil \log_2 C \rceil$ N-dimensional projections (M-dimensional in the CS case). Table II exhibits the underlying motivations related to the proposed hierarchical approach. It clearly show the interest of hierarchical learning in particular when combined with CS. In fact, when learning the hierarchical classifier on CS measurements hardware requirements are dramatically reduced thanks to the signal independent dimensionality reduction performed by CS.

Learning	Memory	Computing
One-vs.-all	NC	$\mathcal{O}(NC)$
One-vs.-one	$NC(C-1)/2$	$\mathcal{O}(NC(C-1)/2)$
Hierarchical	$N(C-1)$	$\mathcal{O}(N \log_2 C)$
Hierarchical+CS	$M(C-1)$	$\mathcal{O}(M \log_2 C)$

TABLE II: A comparison of embedded resources requirements of hierarchical learning approach compared to the one-vs.-all and one-vs.-one strategies.

IV. Simulation results

To evaluate the accuracy (correct predictions over the total number of test samples) and its standard deviation (100 batches) of the proposed methods (*cf.*, Table I), two object classification databases have been used: AT&T database [22] (40 classes, 10 faces per class) and COIL-100 [23] database (32 classes randomly selected out of 100, 72 images per class). In addition, each image is resized to a 32×32 resolution via bicubic interpolation and standardized using the feature scaling method [24] to stay in the scope of a highly constrained image sensor hardware. Two test-benchs are proposed. In the first one, simulations are performed in the signal domain using the original training and test sets (*i.e.*, $\boldsymbol{X} \in \mathbb{R}^{N \times n_1 C}$ and $\boldsymbol{Y} \in \mathbb{R}^{N \times n_1 C}$ respectively). In the second one, simulations are performed in the CS domain using the sensing matrix $\boldsymbol{A} \in \mathbb{R}^{M \times N}$. In this case, the training and test sets are acquired using the matrix $\boldsymbol{A}$ and the hierarchical algorithms are constructed in the CS domain. For the sake of simplicity, simulations are performed using a sensing matrix $\boldsymbol{A}$ following a Bernoulli law with a probability of success of 0.5 [25] with a compression ratio of 25% (*i.e.*, $M = N/4$). We stress that for an embedded application [7] the sensing matrix can be generated "on-the-fly" thanks to digital pseudo-random generators (*e.g.*, LFSR [4], cellular automaton [6]).

In terms of classification accuracy, hierarchical learning yet reduces the average algorithm accuracy by $\approx 3\%$ on original data and $\approx 4\%$ on CS data compared to a straight linear SVM. On the other hand, regarding the decision tree depth (*i.e.*, number of decisions to compute at the inference), our methods deeply outperform the one-vs.-all method, and even more the one-vs.-one. Indeed, in the case of the AT&T, the number of projections to be performed is divided by ≈ 6.6 (6.4 for COIL-100) compared to a one-vs.-all strategy. Finally, it means an equivalent reduction in terms of algorithm complexity and thus power consumption, considering hardware implementation. Note that, the impact of CS is twofold, reducing the size of the SVM coefficients to store and the total amount of multiply-accumulate (MAC) operations to perform on-chip.

V. Conclusion

In this paper, we proposed three balanced clustering methods allowing for hierarchical SVM learning. Simulation results based on two databases show the great interest of our proposed approach in terms of hardware requirements (computational complexity and memory access needs) with an acceptable impact on the classification accuracy. In addition, when combined with CS, the overall memory and on-chip MAC operation needs can even be lowered thanks to signal-independent dimensionality reduction. This paper demonstrates that low-power hardware constrained decision making algorithms can fully take advantage of a hierarchical learning approach combined with CS if the classification accuracy doesn't require to be excessively high (*e.g.*, low-power sensing nodes).

REFERENCES

[1] A. Rocha and S. K. Goldenstein. Multiclass from binary: Expanding one-versus-all, one-versus-one and ecoc-based approaches. *IEEE Transactions on Neural Networks and Learning Systems*, 25(2):289–302, Feb 2014.

[2] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

[3] E. J. Candes and M. B. Wakin. An introduction to compressive sampling. *IEEE Signal Processing Magazine*, 25(2):21–30, March 2008.

[4] L. Jacques, P. Vandergheynst, A. Bibet, V. Majidzadeh, A. Schmid, and Y. Leblebici. Cmos compressed imaging by random convolution. In *2009 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 1113–1116, April 2009.

[5] Y. Oike and A. El Gamal. CMOS image sensor with per-column Sigma Delta ADC and programmable compressed sensing. *IEEE Journal of Solid-State Circuits*, 48(1):318–328, Jan 2013.

[6] W. Guicquero, A. Dupret, and P. Vandergheynst. An algorithm architecture co-design for cmos compressive high dynamic range imaging. *IEEE Transactions on Computational Imaging*, 2(3):190–203, Sept 2016.

[7] W. Benjilali, W. Guicquero, L. Jacques, and G. Sicard. A low-memory compressive image sensor architecture for embedded object recognition. In *2018 IEEE 61st International Midwest Symposium on Circuits and Systems (MWSCAS)*, pages 881–884, Aug 2018.

[8] D. Bankman, L. Yang, B. Moons, M. Verhelst, and B. Murmann. An always-on 3.8 μj/86% cifar-10 mixed-signal binary cnn processor with all memory on chip in 28nm cmos. In *2018 IEEE International Solid - State Circuits Conference - (ISSCC)*, pages 222–224, Feb 2018.

[9] K. Bong, S. Choi, C. Kim, D. Han, and H. J. Yoo. A low-power convolutional neural network face recognition processor and a cis integrated with always-on face detector. *IEEE Journal of Solid-State Circuits*, 53(1):115–123, Jan 2018.

[10] D. Jeon, Q. Dong, Y. Kim, X. Wang, S. Chen, H. Yu, D. Blaauw, and D. Sylvester. A 23-mw face recognition processor with mostly-read 5t memory in 40-nm cmos. *IEEE Journal of Solid-State Circuits*, 52(6):1628–1642, June 2017.

[11] S. Chakraborty, J. W. Stokes, L. Xiao, D. Zhou, M. Marinescu, and A. Thomas. Hierarchical learning for automated malware classification. In *MILCOM 2017 - 2017 IEEE Military Communications Conference (MILCOM)*, pages 23–28, Oct 2017.

[12] V. Vural and J. G. Dy. A hierarchical method for multi-class support vector machines. In *Proceedings of the Twenty-first International Conference on Machine Learning*, ICML '04, pages 105–, New York, NY, USA, 2004. ACM.

[13] J. Yoon, J. Choi, and C. D. Yoo. A hierarchical-structured dictionary learning for image classification. In *2014 IEEE International Conference on Image Processing (ICIP)*, pages 155–159, Oct 2014.

[14] Y. Qu, L. Lin, F. Shen, C. Lu, Y. Wu, Y. Xie, and D. Tao. Joint hierarchical category structure learning and large-scale image classification. *IEEE Transactions on Image Processing*, 26(9):4331–4346, Sept 2017.

[15] N. Keriven, N. Tremblay, Y. Traonmilin, and R. Gribonval. Compressive k-means. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6369–6373, March 2017.

[16] S. Y. Lam and S. K. Choy. K-medians clustering based l1-pca model. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1359–1363, April 2015.

[17] E. J. Candes and T. Tao. Decoding by linear programming. *IEEE Transactions on Information Theory*, 2005.

[18] M. A. Davenport, P. T. Boufounos, M. B. Wakin, and R. G. Baraniuk. Signal processing with compressive measurements. *IEEE Journal of Selected Topics in Signal Processing*, 2010.

[19] S. Afonso, G. Mixon, and B. Recht. Compressive classification and the rare eclipse problem. *arXiv preprint arXiv:1404.3203*, 2014.

[20] V. Cambareri, C. Xu, and L. Jacques. The rare eclipse problem on tiles: Quantised embeddings of disjoint convex sets. In *2017 International Conference on Sampling Theory and Applications*, 2017.

[21] C. D. Martin and M. A. Porter. The Extraordinary SVD. *ArXiv e-prints*, March 2011.

[22] AT&T database. Available [online]: *http://www.cl.cam.ac.uk/research/dtg/attarchive/facedatabase.html*.

[23] COIL-100 database. Available [online]: *http://www1.cs.columbia.edu/CAVE/software/softlib/coil-100.php*.

[24] A. Stolcke, S. Kajarekar, and L. Ferrer. Nonparametric feature normalization for svm-based speaker verification. In *2008 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 1577–1580, March 2008.

[25] L. Jacques and P. Vandergheynst. Compressed sensing: When sparsity meets sampling. Technical report, Wiley-Blackwell, 2010.