Real-time Classification via Sparse Representation in Acoustic Sensor Networks

Bo Wei[†‡], Mingrui Yang[‡], Yiran Shen[†‡], Rajib Rana[‡], Chun Tung Chou[†], Wen Hu[‡]
[†] School of Computer Science and Engineering,
University of New South Wales, Sydney, NSW, Australia
[‡] CSIRO Computational Informatics, Brisbane, Queensland, Australia
{bwei,yrshen,ctchou}@cse.unsw.edu.edu[†]
{mingrui.yang,rajib.rana,wen.hu}@csiro.au[‡]

ABSTRACT

Acoustic Sensor Networks (ASNs) have a wide range of applications in natural and urban environment monitoring, as well as indoor activity monitoring. In-network classification is critically important in ASNs because wireless transmission costs several orders of magnitude more energy than computation. The main challenges of in-network classification in ASNs include effective feature selection, intensive computation requirement and high noise levels. To address these challenges, we propose a sparse representation based featureless, low computational cost, and noise resilient framework for in-network classification in ASNs. The key component of Sparse Approximation based Classification (SAC), ℓ_1 minimization, is a convex optimization problem, and is known to be computationally expensive. Furthermore, SAC algorithms assumes that the test samples are a linear combination of a few training samples in the training sets. For acoustic applications, this results in a very large training dictionary, making the computation infeasible to be performed on resource constrained ASN platforms. Therefore, we propose several techniques to reduce the size of the problem, so as to fit SAC for in-network classification in ASNs. Our extensive evaluation using two real-life datasets (consisting of calls from 14 frog species and 20 cricket species respectively) shows that the proposed SAC framework outperforms conventional approaches such as Support Vector Machines (SVMs) and k-Nearest Neighbor (kNN) in terms of classification accuracy and robustness. Moreover, our SAC approach can deal with multi-label classification which is common in ASNs. Finally, we explore the system design spaces and demonstrate the real-time feasibility of the proposed framework by the implementation and evaluation of an acoustic classification application on an embedded ASN testbed.

Categories and Subject Descriptors

H.4 [**Information Systems Applications**]: Miscellaneous; D.2.8 [**Software Engineering**]: Metrics—*complexity measures, performance measures*

General Terms

Algorithms, Design, Experimentation, Performance

Keywords

ℓ_1 minimization, sparse approximation, audio classification, Acoustic Sensor Networks (ASNs)

SenSys'13, November 11–15, 2013, Roma, Italy.

1. INTRODUCTION

Acoustic Sensor Networks (ASNs) have been prototyped for many military and civilian applications such as animal vocalization recognition [13, 22, 24], military vehicle classification [12, 5], tracking [14] and indoor activity classification [20, 35]. Because the sampling rates of ASNs (e.g., 10kHz or above) are several orders of magnitude higher than those of traditional Wireless Sensor Networks (WSNs) (e.g., 0.1Hz for microclimate sensing applications), enormous amount of data are collected by ASNs. It is known that wireless transmission of a bit costs over 1,000 times more energy than a single 32-bit computation [4]. Therefore, in-network classification, which enables ASN nodes to perform acoustic classification on its own without having to transmit acoustic data, is of critical importance for ASNs.

A popular method for acoustic classification is to first select the appropriate features, which are often problem specific, and then feed the selected features to classifiers such as Support Vector Machines (SVMs) [19, 13, 24]. However, the selection of good features is not a trivial problem and the performance of the classification algorithm often depends on the selection of good features. Moreover, different feature selections can result in significantly different classification accuracy for certain acoustic signals, ranging from 50% to 91.53% [34, 24, 37]. Although Mel-frequency Cepstrum Coefficients (MFCC) is one of the most popular features which, together with SVMs, has been used successfully in ASNs [13, 24, 32], MFCC usually needs other carefully selected features [37] to improve classification performance.

An alternative to feature selection is to use *featureless* classification methods. A recent success story is the Sparse Approximation based Classification (SAC) method proposed by Wright et al. [38] for face recognition. This method forms a dictionary from the pixels of the training images *directly* (i.e., without computing features) and uses ℓ_1 minimization

to determine the best match of the test image among the training classes in the dictionary. The method has been shown to outperform state-of-the-art classifiers based on SVMs and k-Nearest Neighbors (kNN).

Motivated by this promising outcome, we are interested in investigating the use of SAC in ASNs. The biggest challenge is that, due to the high sampling rate of acoustic signals, the dictionary formed by the featureless SAC method is large. This has two important implications for ASNs because this dictionary has to be stored on an ASN node in order to enable in-network classification. First, a large dictionary means that an ASN node can only store a limited number of training classes. This limits the number of different types (or classes) of acoustic signals that an ASN node can classify. Second, the computation time for ℓ_1 minimization is proportional to the size of the dictionary. This means that real-time classification may only be possible if the number of classes is small. The purpose of real-time classification in ASNs is two-fold. First, some ASN applications such as military vehicle tracking [12, 5, 14] require real-time event reporting. Second, a node needs to finish processing the captured audio signal in real-time because the audio signal input is a continuous stream. Otherwise, part of the captured audio signal will be lost.[1] In order to address this problem of high dictionary dimensionality, we propose a novel method to reduce the size of the dictionary without sacrificing classification performance. The reduced dictionary enables each ASN node to complete the classification in real-time on its own. Nodes in ASN are only required to transmit classification results and wireless transmission requirements are significantly reduced. Although each node can perform classification on its own, the network has two important roles to play. First, a network of acoustic sensors is important to increase the size of sensing coverage area. Second, a network is important for ASN re-tasking, e.g. a new dictionary can be transferred to the ASN nodes to enable them to classify new animal species.

The classification problem in ASNs differs from that in many other domains, such as image recognition, in that the test samples in ASNs may include simultaneous sound from multiple sources, e.g., an ASN node may pick up the sound of trains, planes and birds at the same time. This requires multi-label classification, which is a variant of the classification problem where multiple target labels must be assigned to each test instance. We show that our proposed SAC classification framework is able to give a *unified* treatment of the single-label and multi-label classification problems.

Our contributions in this paper are as follows.

- We adopt the SAC framework for acoustic classification. The key advantage of the framework is that it does not require feature selection. The SAC framework does not appear to have been applied to acoustic classification before.
- We propose a novel method to significantly reduce the dimension of the dictionary formed by the SAC method in order to enable real-time classification on resource constrained ASN platforms.
- We propose a novel multi-label classification algorithm for the SAC method.
- We conduct extensive simulations, using two real-life animal vocalization datasets with the sound from 14 frog species and 20 cricket species respectively, to demonstrate the classification accuracy and robustness to environmental noise of the proposed SAC framework. Furthermore, we compare our proposed SAC method against two standard classifiers, SVM and kNN, and show that SAC outperforms SVM and kNN for a wide range of signal-to-noise ratios. Moreover, we compare the multi-label classification accuracy of SAC against a standard classifier ML-kNN and show that SAC gives better performance.
- We explore the system design spaces of SAC and conduct experiments on an outdoor ASN testbed for classifying bird calls. We show that our dictionary reduction method enables real-time classification, and the classification accuracy of SAC is significantly (more than 25%) higher than those of SVM and kNN, which makes SAC a good candidate for the remote ASNs deployed in harsh environment.

The rest of this paper is organized as follows. Section 2 presents the background on the SAC framework proposed in [38]. We then present the details of our proposed SAC framework for ASNs in Section 3. Evaluation results are given in Section 4 (simulation) and Section 5 (outdoor ASN testbed). Related work is presented in Section 6. Finally, we conclude the paper in Section 7.

2. BACKGROUND ON SAC

This section gives an overview of the SAC method presented in [38]. The method solves a single-label classification problem, which aims to return the class that best matches a given test sample. The method assumes that there are s classes indexed by $i = 1, ..., s$. Class i contains t_i training examples. Each training example is assumed to be a column vector with p elements. For example, in [38], which deals with face recognition, each training example contains the intensity levels of the pixels in the training image; therefore, p equals to the number of pixels in a training image. The method is featureless in the sense that the number of elements in a training example is of the same order of magnitude as the amount of raw data describing the example. Note that because training examples and test samples are vectors, we will also refer to them as training vectors and test vectors.

A key idea behind SAC is to assemble all the training vectors from all classes into a *dictionary* matrix. Let $a_{i,j} \in \mathbb{R}^p$ denote the j-th training vector for the i-th class. The dictionary matrix $A \in \mathbb{R}^{p \times n}$ has the form

$$A = [a_{1,1}, \ldots, a_{1,t_1}, \ldots, a_{i,1}, \ldots, a_{i,t_i}, \ldots, a_{s,1}, \ldots, a_{s,t_s}] \quad (1)$$

where $n = \sum_{i=1}^{s} t_i$ is the total number of training vectors. The columns of the dictionary are also known as *atoms*.

In order to explain the classification method, we assume that there is a test vector $y \in \mathbb{R}^p$ belonging to the i-th class. Since y comes from the i-class, ideally, we want y to be dependent on a small subset of training vectors $\{a_{i,1}, \ldots, a_{i,t_i}\}$ in class i only and is independent of the training vectors from all other classes. We can check whether this holds by solving the linear equation

$$y = Ax \quad (2)$$

[1] Queuing theory [1] shows that the size of queue (storage) will not help when arrival rates exceed departure rates.

with unknown vector $x \in \mathbb{R}^n$ where the number of unknowns n in x is equal to the number of columns in the dictionary. If the ideal condition holds, x has the form

$$x_{\text{ideal}} = [0, \ldots, 0, x_{i,1}, \ldots, x_{i,t_i}, 0, \ldots, 0]^T \quad (3)$$

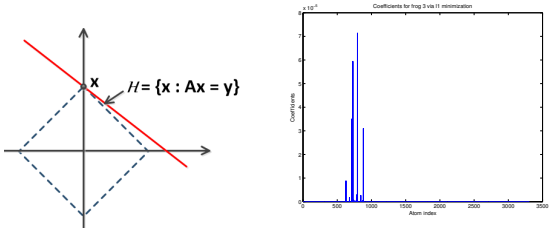
where T denotes matrix transpose. The ideal solution x_{ideal} means that y is a linear combination of the training vectors in i-th class but not others. Furthermore, if y depends only on a small subset of training vectors in class i, only a few of $x_{i,1}, \ldots, x_{i,t_i}$ is non-zero. Therefore, x_{ideal} is a *sparse* vector because most of its elements are zero.

Unfortunately, the ideal condition does not hold because of noise and other reasons. In order to overcome these problems, [38] proposes to solve for x using the ℓ_1 minimization problem:

$$\hat{x} = \arg\min_x \|x\|_1 \quad \text{subject to } \|y - Ax\|_2 < \epsilon, \quad (4)$$

where ϵ is used to account for noise. The optimization objective uses ℓ_1 norm because x_{ideal} is sparse and ℓ_1 norm is known to give a feasible sparse solution compared with other choices of norms [6, 8]. The constraint in (4) is to deal with noise. The optimization problem (4) has been shown to work well even in noisy condition [6].

To demonstrate the intuition why ℓ_1 minimization performs better than the ℓ_2 approaches, we show in Figure 1 and Figure 2 typical ℓ_1 and ℓ_2 solutions to Equation (2) respectively. The ℓ_1 solution is located at the point where $y = Ax$ hits the ℓ_1 ball. For 2-dimensional (2-D) space, the ℓ_1 ball takes the shape of a diamond, see Figure 1(a). In this case, the solution x is located on the axis which means x is spare because it has only one non-zero element. This result can be generalized to higher dimensions. Figure 1(b) shows the solution vector x for classifying *Cyclorana Cultripes* (also known as Grassland Collared Frog). The figure plots x_i against i where x_i is the i-th element of the vector x. It can be seen that x has only a few non-zero elements or is sparse. Figure 2(a) shows the behavior of ℓ_2 solutions in 2-D where the ℓ_2-ball is a circle. The solution x is generally not located on the axis and is therefore not sparse. The ℓ_2 solution of classifying *C. Cultripes* is plotted in Figure 2(b). The solution x is clearly not sparse.

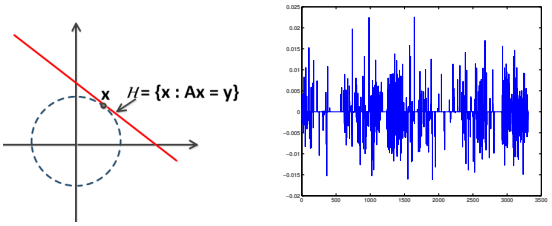


(a) (b)

Figure 1: Sparse solution from ℓ_1 minimization

An issue with (4) is that the dimension p can be large. However, motivated by the results from Compressive Sensing (CS) [3, 6, 8], it is possible to solve a *reduced* dimension minimization problem whose solution is close to that in (4). The reduced dimension problem is:

$$\hat{x}_r = \arg\min_x \|x\|_1 \quad \text{subject to } \|\tilde{y} - \tilde{A}x\|_2 < \epsilon, \quad (5)$$



(a) (b)

Figure 2: Dense solution from ℓ_2 minimization

where $\tilde{y} = Ry$, $\tilde{A} = RA$, and R is an $m \times p$ *random projection matrix* whose elements come from Gaussian, sub-Gaussian or symmetric Bernoulli distribution, and $m \asymp k \log(n/k)$, k being the sparsity of x. Note that the number of rows $(= m)$ in $\tilde{y}$ and $\tilde{A}$ is much smaller than that in the original problem with y and A.

We are now ready to describe the classification algorithm. Assuming that dictionary A and a test vector y are available, and a random projection matrix R has been generated. The first step is to solve (5) and its solution is denoted by $\hat{x}_r$. We can proceed to compute the residuals for each class

$$r_i(y) = \|y - A\delta_i(\hat{x}_r)\|_2, \quad (6)$$

where $\delta_i(\hat{x}_r)$ selects only the nonzero coefficients belonging to class i, i.e., let $\hat{x}_r = [\chi_{1,1}, \ldots, \chi_{1,t_1}, \ldots, \chi_{i,1}, \ldots, \chi_{i,t_i}, \ldots, \chi_{s,t_s}]^T$ then $\delta_i(\hat{x}_r) = [0, \ldots, 0, \chi_{i,1}, \ldots, \chi_{i,t_i}, 0, \ldots, 0]^T$. The chosen class $\hat{i}$ given by

$$\hat{i} = \arg\min_{i=1,\ldots,s} r_i(y), \quad (7)$$

i.e. the class that minimizes the residuals .

3. EFFICIENT ACOUSTIC ℓ_1 CLASSIFIER

In this section, we first discuss the challenges in applying ℓ_1 minimization based classification in ASNs. We then present the architecture of our classification framework and the details of the components.

3.1 Challenges

There have been successful applications of ℓ_1 minimization in many research areas recently. For instance, Wright et al. [38] adopt ideas from sparse approximation and uses ℓ_1 minimization for face recognition. They show that by using ℓ_1 minimization to find the sparse representation, *feature selection is no longer important*. In addition, ℓ_1 minimization is robust to occlusion. This motivates us to investigate the possibility of adopting ℓ_1 minimization for acoustic classification in ASNs.

However, we have to address several issues before it can be applied to ASNs. First, in order for sparsity to hold, the work in [38] requires that each test image must be a linear combination of a *small* number of training images in its class. For acoustic signals, we can use the temporal domain acoustic samples in a time window to form a training vector, and then shift the time window by a fixed *offset* to obtain the next training vector and so on. In order to meet the sparsity requirement in the acoustic domain where a test vector is a

linear combination of a small number of training vectors, the offset has to be made *very small*. We refer to this as the *alignment* problem. The alignment problem means that a dictionary will contain a *large* number of training vectors and this imposes enormous computation burden on the system. Alternatively, we find that if we transform the temporal signals into spectrum domain, then the requirement of alignment (or small offset) can be relaxed significantly. This means that we can reduce the number of training vectors in the dictionary while maintaining classification performance.

Second, despite of its outstanding performance for classification, ℓ_1 minimization problem is known to be expensive to solve in terms of resource consumption. It is a convex optimization problem and an LP solver is needed. In general, each application of an LP solver requires $O(n^3)$ floating-point operations (flops), where n is the number of unknowns. Even for optimized fast ℓ_1 minimization algorithms such as the ℓ_1-homotopy method, the complexity is still $O(k^3 + kmn)$, where k denotes the sparsity of the solution, m is the number of equations, and n is the number of unknowns [10]. Therefore, if we can reduce the number of unknowns, equivalently, the number of atoms/columns in the dictionary, as well as the number of equations in the system, then we will be able to reduce the computational cost, and realize real-time classification on ASNs. Therefore, instead of applying ℓ_1 minimization directly to the classification problem, we first reduce the size of the dictionary to reduce the dimension of the problem, so that it can better fit into the resource constrained ASNs. In Section 3.3.2, we formulate the problem of reducing the number of atoms/columns in a dictionary and show that it is NP-hard. We propose two heuristics to tackle this problem. Note that the work on ℓ_1 classifier has so far been carried out on powerful computer platforms. To the best of our knowledge, our work is the first to consider ℓ_1 classification in resource constrained embedded platforms.

3.2 System Architecture

In this section, we outline the proposed SAC framework for acoustic classification in ASNs. As shown in the flow chart in Figure 3, the whole procedure consists of three parts: offline dictionary learning, online pre-processing of input signals, and online classification.

For the training signal, a silence detection algorithm is applied to the training signal to remove any not-of-interest parts, and a windowed FFT with overlaps is performed to form an initial dictionary. Then a column reduction technique is applied to the initial dictionary to reduce the number of columns in the dictionary (see Section 3.3) which is then followed by a multiplication of a projection matrix to reduce the number of rows in the dictionary. We denote the reduced training dictionary by matrix $\tilde{A}$.

After the acquisition of the test signal, we again apply silence detection to only keep the interesting parts. A windowed FFT with no overlap is then applied to the retained sound segment. The same projection matrix (as used for training) is used to reduce the dimension of the test signal and provide the measurement vector $\tilde{y}$, see Section 3.4.

Now both the training dictionary $\tilde{A}$ and the measurement $\tilde{y}$ are passed to the classifier. The ℓ_1 classifier first finds the sparse coefficient vector x. It then calculates the residuals for each class i in the training dictionary and computes a threshold. If the residual for class i is less than the threshold, then a class i in the test signal is identified. Otherwise, if all residuals are greater than the threshold, then the test signal is an unknown class. See Section 3.5 for more details.

3.3 The Training Dictionary

Obtaining the right dictionary is always one of the most important steps for classification problems because of its effect on classification accuracy. For different applications, the dictionary needs to be trained accordingly.

3.3.1 Silence Removal and Segment Formation

We first describe how the *initial* training dictionary is obtained for our classification scheme. For the training sound samples from each class, we first apply a simple silence removal technique as shown in Algorithm 1, where the threshold ρ is learned from the environment. RMS in line 3 of Algorithm 1 stands for root mean squares.

The signal is then partitioned into small segments using fixed window size and overlap size. Let w and o denote, respectively, the window size and overlap size, in number of samples. The first segment consists of samples 1 to w, second segment consists of samples $w - o + 1$ to $2w - o$ (i.e., shifting $w - o$ samples to create an overlap of o samples), third segment consists of samples $2(w-o)+1$ to $3w-2o$ and so on. Note that the overlap is required so that a test vector can be expressed as a linear combination of a small number of training vectors. As discussed in Section 3.1 on the alignment problem, if training vectors are formed from temporal samples, then a large overlap (or small offset) is required. However, only a small overlap (or large offset) is required if training vectors are formed from spectrum domain data.

For each segment, a windowed FFT is performed and the spectrum energy (i.e. the magnitude of the FFT coefficients) is calculated to give an atom (column) of the dictionary. The atoms of all the classes are then put together to form the initial training dictionary A where the atoms from each class is a sub-matrix of A. The class boundary information for each class in A is also recorded. Note that the number of elements in each atom is of the same order of magnitude as the amount of data in a segment, therefore, the method is featureless.

Algorithm 1 Silence Removal

1: **Input**: Audio Segment $SG_{i=1:Z} \in \mathbb{R}^{\alpha} > 1$, where Z is the total number of segments and ρ is the threshold
2: **for** $i = 1 : Z$ **do**
3: **if** RMS $(SG_i) < \rho$ **then**
4: Remove (SG_i)
5: **end if**
6: **end for**

3.3.2 Column Reduction

As mentioned in Section 3.1, one of the major factors that affects the speed of ℓ_1 minimization is the number of unknowns in the system, or equivalently, the number of atoms in the dictionary. When inspecting the initial training dictionary obtained above, we notice that many of the atoms within each class are highly correlated. This means that the atoms of each class contain a lot of redundant information, which is unnecessary for representing the class, and

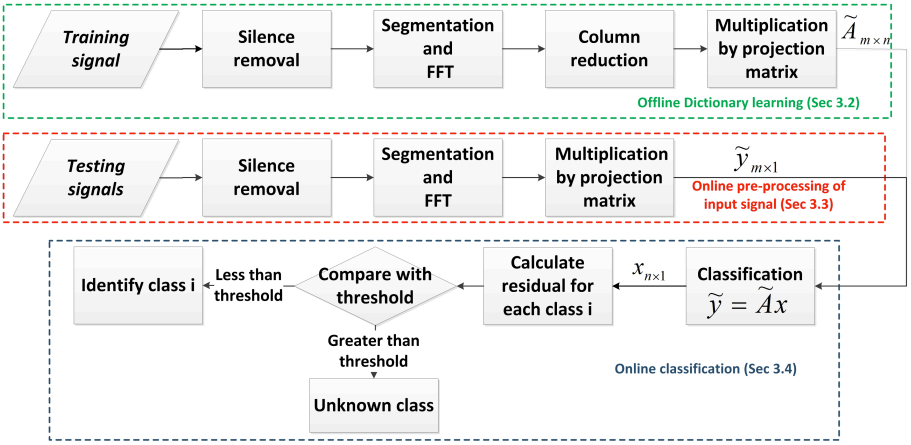


Figure 3: System flow chart

hopefully can be removed. This motivates us to reduce the number of atoms within each class.

In order to reduce the redundancy in each class, we need a criterion to measure the redundancy. As a first step, we first normalize the columns of the dictionary A so that each column has a unit norm. As a result, the inner product between two training vectors in the dictionary is equal to the cross-correlation coefficient between them. We define the *mutual coherence* $M(A)$ of a dictionary A as the maximum magnitude of the cross-correlation coefficient of all possible pairs of columns in A. Mathematically, let a_i, $i = 1, \ldots, n$, denote the columns of a matrix $A \in \mathbb{R}^{p\times n}$ with $\|a_i\|_2 = 1$. The mutual coherence $M(A)$ of A is:

$$M(A) = \max_{i\neq j} |\langle a_i, a_j\rangle|, \tag{8}$$

where $\langle\cdot,\cdot\rangle$ denotes the inner product. Clearly, the smaller $M(A)$ is, the less redundancy the dictionary A has. The concept of mutual coherence is also used in CS to measure how well the minimization problem (4) can solve a sparse problem [9, 11, 10] and is therefore a relevant measure.

We propose to reduce the number of columns of the initial dictionary by reducing the mutual coherence for each class. In other words, within each class, we want to find a subset of atoms with a certain cardinality whose mutual coherence is minimized over all possible combinations. This can be formulated as the following optimization problem

$$\arg\min_{I} \max_{i,j\in I, i\neq j} |\langle a_i, a_j\rangle| \quad \text{subject to } |I| = \tau, \tag{9}$$

where I denotes an index set and τ is the cardinality of I, which is given. However, this problem is known to be NP-hard [25]. That is, in the general case, no known efficient algorithm can solve this optimization problem exactly in polynomial time. It is then natural to seek for an alternative heuristic approach: greedy algorithm.

Greedy Algorithm

We propose here a greedy algorithm to reduce the redundancy of the training dictionary. Our greedy approach can be described as follows. For each class i, we randomly select one atom from that class as an initial atom (Line 2 of Algorithm 2). In the following iterations, always pick one atom from the remaining atoms that minimizes the maximum of the corresponding mutual coherence between this atom and the already selected ones (Line 6). If the desired number of atoms is reached or the preset threshold for coherence is satisfied (Line 3 and Line 4), stop the iteration and continue to the next class. By applying this approach, we are able to remove significant amount of redundant information and reduce substantially the number of atoms in the training dictionary, and therefore speed up the ℓ_1 minimization part for classification significantly.

Algorithm 2 Greedy Training Dictionary Column Reduction

1: **for** each class i **do**
2: Input: Dictionary A_i for class i, initial $A_{I_i} = \{a_1\}$, $A_{J_i} = A_i \setminus \{a_1\}$, where a_1 is a column chosen randomly from A_i.
3: **for** $j <$ given number of columns **do**
4: **if** $\max_{a_k\in A_{I_i}, a_j\in A_{J_i}} |\langle a_j, a_k\rangle| > threshold$ **then**
5: //a_k, a_j are columns in A_{I_i} and A_{J_i} respectively
6: $\tilde{a}_j = \arg\min_{a_j\in A_{J_i}} \max_{a_k\in A_{I_i}} |\langle a_j, a_k\rangle|$
7: //a_j and a_k are column in A_{J_i} and A_{I_i} respectively
8: $A_{I_i} = A_{I_i} \bigcup \{\tilde{a}_j\}$
9: $A_{J_i} = A_i \setminus A_{I_i}$
10: **else**
11: break
12: **end if**
13: j++
14: **end for**
15: Output: Reduced dictionary A_{I_i} for class i.
16: **end for**

Tabu Search

However, greedy algorithms are known to converge only to the local optimum, which heavily depends on the initial guess. This local optimum could be way far from the global optimum. Therefore, to ensure our greedy algorithm is giving us the good quality result, we compare it with the *tabu search* algorithm [16].

Tabu search is a meta-heuristic algorithm for searching solutions to combinational optimization problems, and was

widely used to produce good quality approximation results for NP-hard problems efficiently [2, 23, 27]. Tabu search enhances the performance of local search algorithms by exploring unreached areas of the greedy algorithms. The computational complexity of tabu search is $O(n^2)$ which means it can be solved in polynomial-time. It utilizes memory structures by defining the neighborhood of the current solution and two tabu lists. The size of the neighborhood can be tuned according to the processing capability of the devices, and can be as large as the size of the search space. The tabu lists store elements according to specific rules. For instance, in our case, one tabu list stores the column indices that cannot be visited within a certain number of iterations, and the other list stores the column indices that cannot be removed from the solution space within another number of iterations. The number of iterations are determined by the size of the tabu lists, which should be large enough to avoid cycles. We tune them as half of the size of the search space and the solution space respectively. Details of the tabu search algorithm are shown in Algorithm 3.

Algorithm 3 Tabu Search for Column Reduction

for each class i **do**
  Input: Dictionary A_i and number of columns of output class t_i;
  Initialization: allocate two empty lists: L_{in} and L_{out}, stop criteria $stop\_i = 500$; $BestCoh = Inf$; $i = 0$,
  while $i < stop\_i$ **do**
    $CurrentCoh = Inf$, $j = 0$
    Randomly choose t_i columns from A_i to form $\hat{\mathcal{P}}_{t_i}$, record the indices list of $\hat{\mathcal{P}}_{t_i}$ as $\hat{L}$ ($\hat{L} \cap L_{out} = \emptyset$; $\hat{L} \cap L_{in} = L_{in}$); The lists are first-in-first-out;
    while $j < t_i$ **do**
      $\hat{\mathcal{P}}_{t_i-1} = \hat{\mathcal{P}}_{t_i} \backslash \{\hat{\mathcal{P}}_{t_i}\{j\}\}$;
      $A_c = A_i \backslash \hat{\mathcal{P}}_{t_i-1}$;
      $Coh_j = min_{a_d \in A_c} \max_{p_k \in \hat{\mathcal{P}}_{t_i-1}} |p_k^T a_d|$
    //a_d and p_k are column in A_c and $\hat{\mathcal{P}}_{t_i-1}$ respectively
      if $Coh_j < CurrentCoh$ **then**
        $CurrentCoh = Coh_j$; $PotentialObj = \hat{\mathcal{P}}_{t_i}\{j\}$
        Store current solution $\hat{\mathcal{P}}_{t_i}$ in $\bar{\mathcal{P}}$;
      end if
      j++
    end while
    if $CurrentCoh > BestCoh$ **then**
      $BestCoh = CurrentCoh$; $BestSubmtx = \bar{\mathcal{P}}$
      Clear L_{out}; Push $PotentialObj$ in L_{in}
      $i = 0$
    else
      Push $\hat{\mathcal{P}}_{t_i}$ in L_{out}
      i++
    end if
  end while
  Outputs: Reduced dictionary $BestSubmtx$ for class i
end for

This algorithm aims to pick t_i feasible columns for class i in the search space to minimize the mutual coherence within each class in the training dictionary.

3.4 Pre-processing of the Test Signal

The construction of the testing set is similar to that of the training dictionary except that no overlap is performed for the purpose of real-time online classification. For an incoming testing signal, we first run Algorithm 1 to remove the uninteresting parts. Then a windowed FFT, of the same length as used in training, is applied to obtain segments in spectrum domain. The test vectors, which consist of the magnitude of the FFT coefficients, are then passed to the classifier for classification.

3.5 The multi-label ℓ_1 Classifier

Recall that the classification problem in ASNs is of multi-label type, here we describe how ℓ_1 minimization can be used to perform single-label and multi-label classification in a unified manner. We assume the following is available: A dictionary $A \in \mathbb{R}^{p \times n}$ which has already gone through column reduction to reduce mutual coherence, a test vector $y \in \mathbb{R}^p$ and a projection matrix $R \in \mathbb{R}^{m \times p}$. We first solve the minimization problem (5). The dimension of this optimization problem has been reduced because the $\tilde{A} = RA$ matrix has a reduced number of rows by using projection matrix R and a reduced number of columns due to column reduction. Note that in machine learning literature, the rows of the matrix $\tilde{A} = RA$ are interpreted as features. In our case, $\tilde{A}$ is obtained from the multiplication of a random matrix R with the dictionary A. No elaborate computation processes are used to construct $\tilde{A}$, or in other words, our method does not require careful feature selection.

After solving (5), the next step is to compute the residuals of each class according to Equation (6). Again, let $r_i(y)$ denote the residuals of class i. Let μ and σ denote, respectively, the mean and standard deviation of $r_1, \ldots, r_s$. *If $r_i < \mu - 2\sigma$ for some class i, then a match is identified.* By applying this technique, we can reduce the number of false positives. In particular, it is useful for avoiding misclassification when there are unknown classes recorded as test signals which do not belong to any of the known training classes, because the solution will give relatively even residuals for all the classes.

This setting can also deal with the case when multi-label sounds are captured within a same time window by identifying all the classes i's whose residuals r_i's are less than the threshold $\mu - 2\sigma$. Figure 4 shows an example of the residual plot for classifying two types of frogs simultaneously. It is clear from the plot that the residuals for both classes are below the threshold. Therefore, both types of frogs can be correctly identified.

4. PERFORMANCE EVALUATION

In this section, we evaluate the performance of our proposed SAC based acoustic classification framework via simulation.

There are two datasets we use for our evaluation. The first one contains sound samples recorded from fourteen different species of frogs, for both training and testing. For completeness, they are *Cyclorana Cryptotis*, *Cyclorana Cultripes*, *Limnodynastes Convexiusculus*, *Litoria Caerulea*, *Litoria Inermis*, *Litoria Nasuta*, *Litoria Pallida*, *Litoria Rubella*, *Litoria Tornieri*, *Notaden Melanoscaphus*, *Ranidella Bilingua*, *Ranidella Deserticola*, *Uperoleia Lithomoda*, and *Bufo Marinus*. The sampling frequency for this dataset is at 24kHz. The second dataset is from [21], which is available at `http://www.cs.ucr.edu/~yhao/animalsoundfingerprint.`

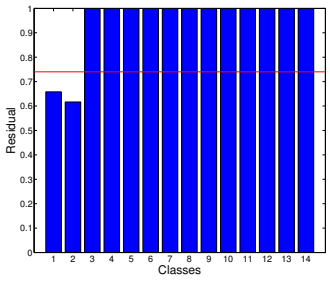


Figure 4: Residual plots demonstrating multi-label classification. The horizontal line shows the threshold. Both classes 1 and 2 have correctly been identified.

`html`. It contains sound samples from twenty different species of crickets. The silence removal algorithm (Algorithm 1) is applied to the raw signals, so as to remove uninteresting parts. We obtain 228 event segments from frog dataset and 663 event segments from cricket dataset respectively.

4.1 Goals, Metrics and Methodology

Our goal in this evaluation is to show that our acoustic classification scheme is featureless, accurate, and robust. For this purpose, we evaluate the performance of our classification scheme on two different datasets with respect to different signal-to-noise ratios (SNRs). In addition, we compare our proposed approach against the conventional approaches, kNN and SVM, and show that our scheme performs better. We also demonstrate that our proposed framework is feasible for multi-label sound classification scenario.

A common approach for evaluating the performance of a classifier is to first partition the dataset into complementary subsets, then perform the analysis on one subset (the training set), and validate on the other subset (the testing set). A generalization of this method is called the *K-fold cross-validation*. The data set is first partitioned into K disjoint subsets of roughly equal size. Then the classifier is evaluated over K rounds. For each round, a different subset is retained as the testing set, while training is carried out on the other $(K-1)$ subsets. The validation results are then averaged over the rounds. In our simulation, we use $K = 3$, the 3-fold cross-validation to obtain the training and testing sets, which are used to evaluate the ℓ_1 classifier, the kNN classifier, and the SVM classifier. For the ℓ_1 classifier, we use ℓ_1-homotopy [29] as the ℓ_1 minimization algorithm. The kNN classifier is directly from Matlab. And the SVM classifier we use is from [7].

The metric we use to evaluate the performance of all the classifiers for single-label classification is the accuracy, which is simply the number of true detections over the total number of test vectors. Since random projection matrices are used, we run the simulation ten times. The accuracy is first averaged over these ten runs, and then averaged over the three folds. In the results, we show both the averaged accuracy and its standard deviation. For the multi-label case, Hamming loss is used as the performance metric. Hamming loss is a typical metric used for multi-label classification [30, 41], which is the average of the symmetric difference of the outcome index set and the ground truth index set over all the testing vectors and training classes.

To demonstrate the robustness of our scheme, we add in environmental noise, which is scaled to different magnitudes, to our testing signals to create different SNRs. The environment noise is from the same national park where the frog calls are recorded. We compare our framework against kNN and SVM with respect to different SNR levels.

For illustration purpose, Figure 5 shows some sample sound segments from our simulation and experiment. The first row is in spectrum domain, and the second row is in temporal domain. Plot (a)-(c) show sample calls of frog *C. Cryptotis* with no noise, with 0dB SNR, and with 10dB SNR respectively. Plot (d) shows the sample call of another frog *C. Cultripes* with no noise. Plot (e) shows the combination of the calls from these two frogs with 0dB signal-to-signal ratio (SSR) for multi-label classification in Section 4.4. Plots (f) and (g) show typical cockatoo and rainbow lorikeet calls from our outdoor experiment (discussed in Section 5), where the noise level is between that in (b) and (c).

4.2 Performance of SAC

We first evaluate the performance of our framework in a single-label multi-class classification setting. The initial training dictionary is obtained from windowed FFT with an overlap ratio (i.e. ratio of overlap size to window size) of 15/16. Figure 6(a) shows the classification accuracy for different SNRs with five different window sizes from 2^{10} to 2^{14} samples, which is the largest possible number of samples for the training signals. To avoid possible effects from other factors, we do not use projection matrix or column reduction to reduce the dimension of the dictionary for Figure 6(a). The figure shows a clear performance gain as we increase the window size. Thus, from now on, we fix the window size to be 2^{14} samples.

4.2.1 Performance on Frog Data

In this section, we use Rademarcher (symmetric Bernoulli) matrix as the random projection matrix because of its efficient implementation. The number of random projections, which equals to number of rows in the projection matrix, should be of the order $k \log(n/k)$ according to compressive sensing theory. However, in real applications, most of the time it is impossible to show that the assumptions for this bound are satisfied. Figure 6(b) shows the impact of the number of projections on classification accuracy at different SNR levels with no column reduction for the frog data. It shows that classification accuracy increases with the number of projections but the improvement diminishes when the number of projections is large. This observation can be explained by the fact that the number of projections plays the same role as the number of features. However, a significant difference of the ℓ_1 method is that it uses random matrices to generate "features", which means careful feature selection is not required. When the number of projections is 300, the classification accuracy has already reached 95% at 2dB SNR, and there is no significant improvement in accuracy if we keep increasing the number of projections. Therefore, from now on in this simulation, we fix the number of projections to be 300 if not specified for the frog data.

Figure 6(c) investigates the effect of column reduction using Algorithm 2 on classification accuracy at different SNR

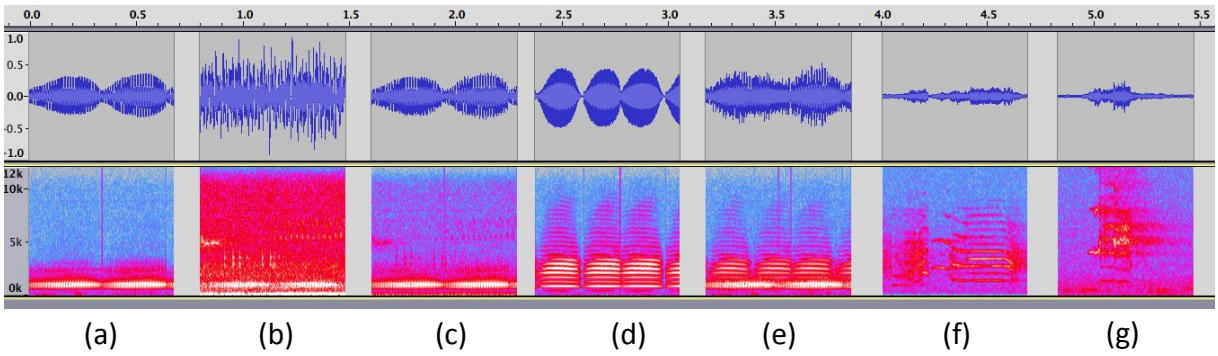
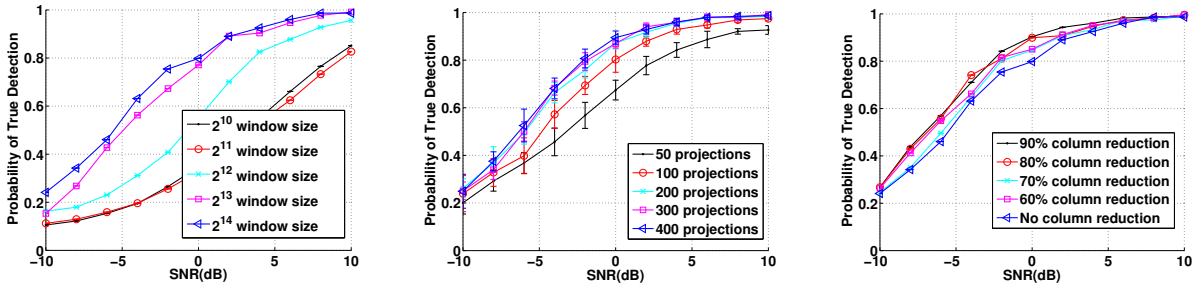


Figure 5: (Picture view best in color) (a) *C. Cryptotis'* call without noise, (b) *C. Cryptotis'* call with noise (0dB SNR), (c) *C. Cryptotis'* call with noise (10dB SNR), (d) *C. Cultripes'* call without noise, (e) Combined sound from (a) and (d) (0dB SSR), (f) Typical cockatoo sound from experiment, (g) Typical rainbow lorikeet call from experiment



(a) For varying number of window size (b) For varying number of projections (c) For varying number of columns

Figure 6: Frog: Accuracy versus SNR curves.

levels. The initial training dictionary has 2,487 atoms, and the test set has 76 test vectors. We compare five different column reduction strategies: 90% (i.e.,10% of the columns are retained), 80%, 70%, 60% reduction rate, and no reduction. Notice that we do not use projections for this plot so as to avoid possible interference from other factors. We observe that we can reduce as much as 90% of the columns with little impact on classification accuracy at every SNR level. Therefore, from now on, we take 90% column reduction rate as our default setting for evaluations.

As discussed in Section 3, we need to compare the results from our greedy algorithm with that of the well known mega-heuristic approach, tabu search. Figure 7(a) shows the comparison of these two algorithms. We see no difference between these two algorithms. Therefore, the performance of our greedy algorithm is close to optimal. For fair comparison, equal reduction rate of 90% is used for each class in the greedy algorithm, and tabu search algorithm. Note that we have tried other column reduction rates, which show similar phenomena.

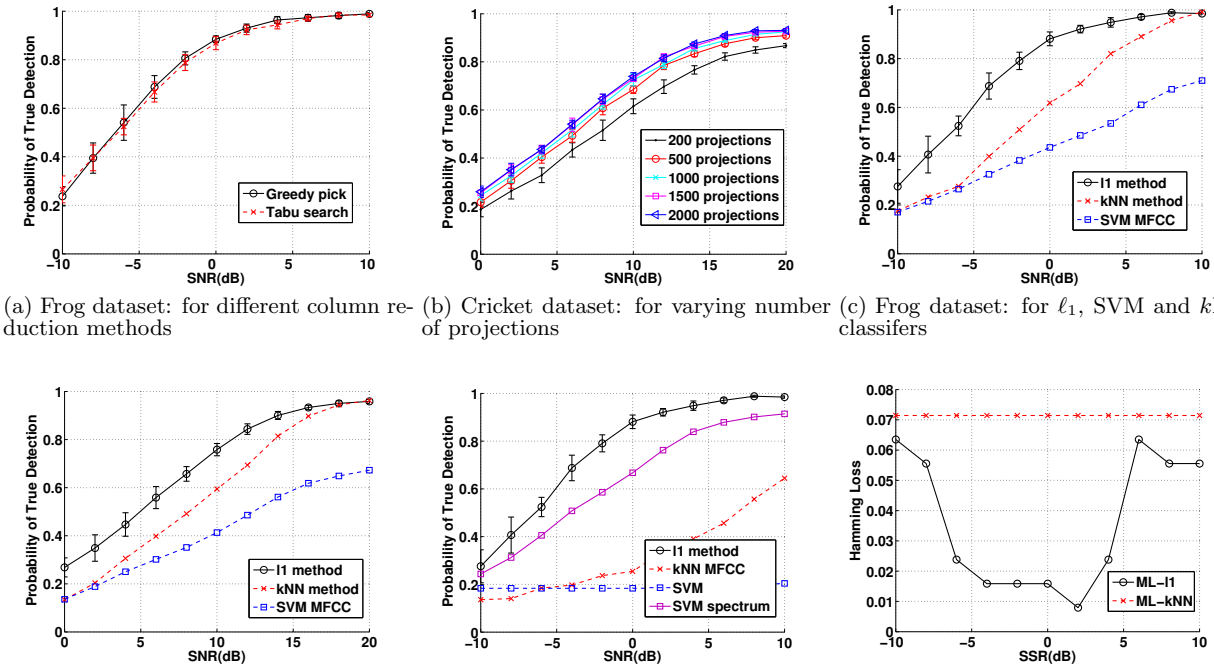
4.2.2 *Performance on Cricket Data*

Similar to the frog data, Figure 7(b) shows the impact of different number of projections on the classification accuracy for the cricket dataset with different SNR levels. To avoid interference by other factors, no column reduction is used. The figures shows that 200 projections give the worst performance in terms of classification accuracy. But again, as the number of projections increases, the improvement on the classification accuracy diminishes. There is no more significant difference when the number of projections exceeds 1,000. This experiment demonstrates that the number of projections required depends on the dataset.

4.3 Comparing to Benchmarks

In this paper we employ both SVMs and kNN to benchmark the classification performance of our proposed SAC framework for ASNs. We choose these two classifiers due to their wide acceptance for various audio processing, including phonetic segmentation, audio classification, speech recognition etc.

For SVM, we choose MFCC as the feature because it is the most popular choice for representing audio signals. We calculate the MFCC of each window by transforming the power spectrum of each window into the logarithmic mel-frequency spectrum. The SVM is trained using the MFCC extracted from different training classes, and classifies the testing signal according to its MFCC features. For kNN,



(a) Frog dataset: for different column reduction methods

(b) Cricket dataset: for varying number of projections

(c) Frog dataset: for ℓ_1, SVM and kNN classifers

(d) Cricket dataset: for ℓ_1, SVM and kNN classifers

(e) Frog dataset: for kNN and SVM with other settings

(f) Frog dataset: performance of multi-label classification

Figure 7: Accuracy versus SNR curves.

we use similar settings as for our ℓ_1 classifier, e.g., sound signals in power spectrum domain, because the employment of MFCC in kNN produces bad result (discussed with Figure 7(e) later). Note that we do not perform matrix column reduction and random projection for kNN because it produced significantly worse result. Therefore, the dimension of kNN is significantly (more than 150 times) larger than that of ℓ_1.

Figure 7(c) and Figure 7(d) compare the performance of our ℓ_1 classifier to SVM with MFCC and kNN for, respectively, the frog dataset and the cricket dataset. For the frog dataset, SVM with MFCC performs the worst among all the three classifiers. This suggests that the different classes are not well separated in the feature space. Therefore, a small amount of noise can cause erroneous classification by SVM. This also explains why MFCC usually needs other carefully selected features [37] to improve classification performance. Our ℓ_1 classifier performs much better than the other two classifiers, especially when SNR is low. For instance, for the frog data, when SNR is 0dB, the ℓ_1 classifier is at least 20% more accurate. The ℓ_1 classifier is more robust than kNN because a test vector is "compared" against all training vectors (see Eq (2)) in case of ℓ_1 but is only compared to individual vectors for kNN. For the cricket data, SVM with MFCC still has the worst performance, while the ℓ_1 classifier and the kNN classifier have similar performance when the SNR is high. However, when SNR is low, our ℓ_1 classifier still outperforms the kNN classifier significantly (more than 15% when SNR is smaller than 10dB). These results show that, besides the featureless property, the ℓ_1 classifier is also robust to noise compared to the conventional approaches, which coincides with our reasoning in Section 2.

Clearly, the robustness to noise makes our ℓ_1 classifier more suitable for real deployments in noisy environments. Furthermore, the better classification performance in low SNR means a larger monitoring coverage, since sound samples from distance typically have low SNR. Alternatively, it means that less sensors are needed for monitoring a certain area of interest.

For the purpose of completeness, we have also evaluated the performance of SVM in temporal domain, SVM in spectrum domain, and kNN with MFCC. They all perform worse than the ℓ_1 classifier. The results are shown in Figure 7(e) for the frog dataset. Note that the ℓ_1 classifier's performance curve in Figure 7(e) is identical to that in Figure 7(c).

4.4 Multi-label Classification

As mentioned in Section 3.5, the scheme we propose is able to handle multi-label classification problems. The metric for multi-label classification is much more complicated than that for the single-label case. Recall from Section 4.1, we will use Hamming loss as the performance metric. Hamming loss has a range from 0 to 1. With a Hamming loss value 0, a classifier has perfect classification performance on multi-label scenario; and with a value 1, it totally fails.

To demonstrate the performance of our proposed framework on multi-label classification, we manually mix the calls from two types of frogs, *C. Cryptotis* and *C. Cultripes*. We vary the relative strengths of their calls to create different signal-to-signal ratios (SSRs), defined in a way similar to SNR. We vary the SSRs between -10dB to 10dB. At the ex-

treme SSRs of ±10dB, the signal energy of one frog call is 10 times that of the other. When the SSR is 0dB, it means that the energy of the two calls are equal.

As a benchmark, we compare the result of ML-kNN classifier [41] with our ℓ_1 classifier. We train the ML-kNN classifier with both the smoothness parameter and the number of neighbors set to one. Figure 7(f) shows the Hamming loss curves with varying SSRs for both ℓ_1 and ML-kNN classifiers. If the SSR is either very low or very high, the two classifiers have similar performance, where our ℓ_1 classifier is still a bit better. However, when the SSR is around 0dB, which means the two classes have about the same energy, our ℓ_1 classifier outperforms the ML-kNN classifier significantly.

This characteristic of our ℓ_1 classifier could be very useful in practice. It provides the ASN deployment the ability to detect multiple sound sources when they occur at the same time, especially when the energy levels of the sound sources are not very different from each other.

5. EXPERIMENTS ON TESTBED

In this section, we evaluate the performance of the proposed SAC on an outdoor ASN testbed. The testbed, which is located on our campus with thin vegetation, consists of five nodes (Figure 8(a)) configured as *Ad-hoc* mode with a star network topology. The aim of the experiments is automatic bird vocalization recognition, which is a typical pattern recognition problem [32, 13, 37]. We choose two bird species that are frequently observed on our campus: cockatoos (Figure 8(b)) and rainbow lorikeets (Figure 8(c)). The experiment is a multi-class classification with 3 classes: the calls of cockatoos, the calls of rainbow lorikeets and environmental noise (which includes the sound all other birds, crickets etc.). The goal of the experiment is to demonstrate that the reduced dictionary enables each ASN node to complete the real-time classification on its own. After classification, each node sends its classification results to a server.

5.1 System Description

Table 1: Node Power Consumption at 5V

Module	Consumption (W)
CPU	2.05
CPU + microphone	2.1
CPU + Wifi (idle)	2.45
CPU + Wifi (Rx)	2.67
CPU + Wifi (Tx)	2.78

The nodes in our ASN testbed are based on the Pandaboard ES (see the left side of Figure 9), a single board computer costing US$182, which has a 1.2GHz ARM Cortex-A9 with 1GB of RAM and a 4GB SD-card. Pandaboard also features an 802.11 interface for wireless communication, and runs Ubuntu Linux distribution. Each node hosts 2 USB ports, one of which connects to a USB microphone. The microphones are configured to sample at 24kHz. We implemented the ℓ_1 *Homotopy* — a fast ℓ_1 minimization algorithm which is also used in simulation experiments in Section 4 — in C++ based on GNU Scientific Library (GSL) for the Pandaboard platform.

A node is powered by a 12V 7.2Ah rechargeable battery, and an optional 5W, 12V solar panel (see the right side of

(a)

(b)

(c)

Figure 8: (a) Star topology of the 5 deployed nodes. (b) A picture of cockatoo. (c) A picture of rainbow lorikeet.

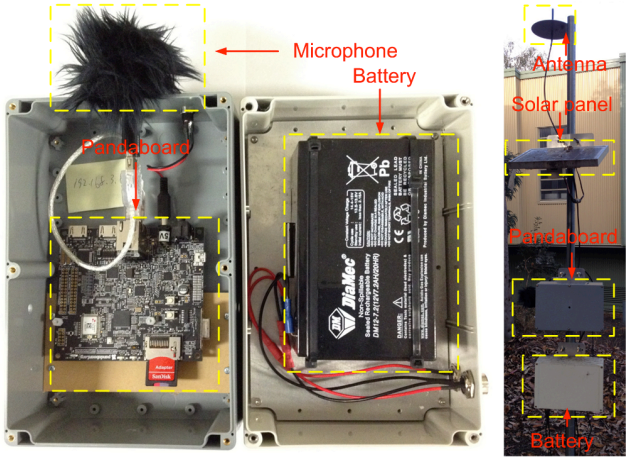


Figure 9: A Pandaboard node (left), and a deployed node (right)

Figure 9). Table 1 shows the power consumption of different modules. The nodes in SolarStore testbed [40] consume 10W (low load) and 15W (high load) energy respectively. Therefore, the nodes in our ASN testbed is approximately 3.5 to 5.4 times more energy efficient. Without solar panel, a node in our ASN testbed will run continuously for more than 31 hours, which is significantly longer than the previous platforms such as ENSBox [15]. A solar panel with 8-hour exposure to direct sunlight per day can maintain a 50% duty cycle at 85% solar charge efficiency.

The nodes use Network Time Protocol (NTP) for time synchronization. We use one node as the NTP server, and the other nodes as the NTP clients. The NTP clients send request for time synchronization every 10 seconds. The time synchronization accuracy is 25 ms in average.

We introduce two components to counter one of the main challenges for outdoor ASNs: wind sound. Firstly, we install foam and fur windscreen around each microphone (see the left side of Figure 9), which can reduce the effect of wind sound significantly. Secondly, we apply a fifth order Butterworth high pass filter to cut off the recorded audio frequency lower than 200Hz, because most of the wind audio energy is in the frequency band below 200Hz, and most of the vocalization energy of the cockatoo and rainbow lorikeet is in the frequency band higher than 200Hz.

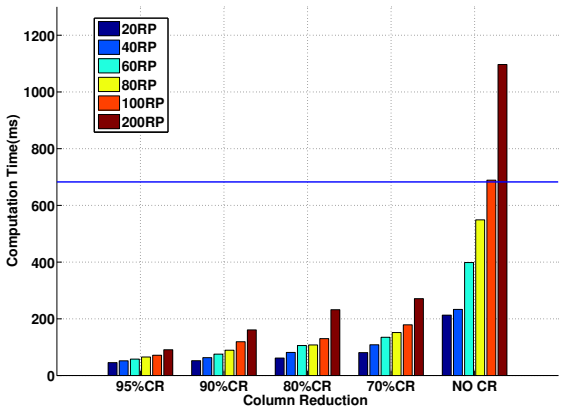
5.2 Classifier Parameter Selection

The proposed ℓ_1 classifier has two important parameters — the percentage of column reduction and the number of projections — which have to be tuned for each classification problem. We tune these parameters by using training data sets collected from our experiment testbed. Altogether 609 pieces of sound recordings with length of $2^{14}/24,000 = 682.67$ms have been collected. We then process these sound recordings using the procedure described in Section 4 to obtain an initial, but rather redundant, dictionary. The initial dictionary contains 313 and 231 columns, respectively, for cockatoos and rainbow lorrikeets.

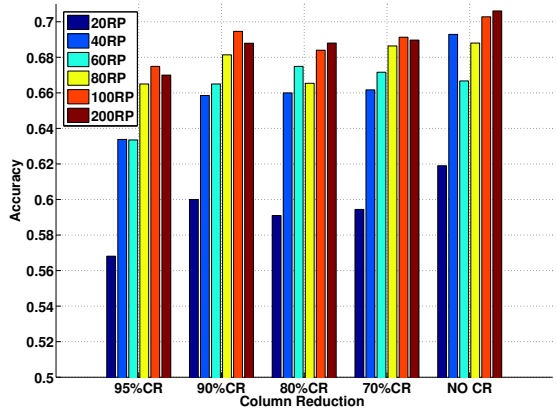
With this initial dictionary, we test the impact on the classification accuracy and computation time by varying the percentage of column reduction and number of projections. We use 5 different column reduction percentages: no reduction, 70%, 80%, 90% and 95%; and 6 different number of projections: 20, 40, 60, 80, 100 and 200. The accuracy is evaluated by simulations, similar to those described in Section 4. The computation time is evaluated on Pandaboard. The results on classification accuracy and computation time and are shown, respectively, in Figures 10(a) and 10(b).

Figure 10(a) shows the column reduction can significantly shorten the computation time. For example, with 100 random projections, the computation time reduces by a factor of more than 5 when 90% column reduction is used. Figure 10(b) shows that the classification accuracy increases with larger number of projections but the improvement diminishes when more than 100 random projections are used. Based on this observation, we choose 100 projections as the parameter. With 100 projections, we see that 90% column reduction can give almost the same classification accuracy as 70% or 80% column reduction, so we select 90% column reduction as the parameter.

Pandaboard is a rather powerful embedded platform with 1 GB RAM and a 1.2 GHz micro-controller. A node with less RAM and slower micro controller cannot process such a large matrix (with 90% column reduction and 100 projections), and will result in less classification accuracy or cannot process all the real-time captured audio signals (only a percentage of captured signals will be processed). This is a design space of the the ASN application developers. For example, if an application developer has an ASN platform which is approximately 50% of Pandaboard's processing power, he/she can choose either process *all* the capture audio signals with 95% column reduction and 40 projection, or process *half* of the capture signals with 90% column reduction and 100 projections. The first option will achieve higher classification accuracy (approximately 69% as showed in Figure 10(b)) than the second option (approximately 63% as showed in Figure 10(b)).



(a)



(b)

Figure 10: (a) Computation time. (b) Classification accuracy. CR = Column Reduction, RP = Random Projection.

5.3 System Performance

In order to realize real-time in-network classification, an ASN node needs to complete the classification of a time window of data within the duration of the window. Table 2 shows the computation time (in milliseconds) of the major modules of the proposed SAC framework for two scenarios: No column reduction with 100 projections and 90% column reduction with 100 projections. The table shows that ℓ_1 *minimization* is indeed computation intensive and consumes most of the Pandaboard CPU time. Without column reduction, the *residual calculation* and ℓ_1 *minimization* module consume approximately 75% of the CPU time. The proposed *90% Column Reduction* improves the speed of these

two modules by approximately six times. Because the duration of each time window is 682.67ms and the total process time is approximately 120ms, the nodes in our ASN testbed can process all the captured acoustic data in real time.

Table 2: Mean (Standard Deviation) of Computation Time (ms). CR – Column Reduction.

Module	No CR	90% CR
Silence removal	20.38(2.04)	
16384-point FFT	15.33(0.63)	
Random projection	26.41(0.77)	
ℓ_1 minimization	534.71(184.83)	46.16(15.20)
Residual	92.08(2.37)	10.32(0.49)
Total time	688.91	118.60

5.4 Outdoor Validation Experiments

After tuning the classifier parameters, we implement the classifier on the ASN testbed. Each node in the testbed runs the proposed classifier locally produces classification results for all the time periods that are not silent. At the same time, the nodes store the audio records of non-silent periods in the SD cards for ground truth and comparison purposes.

The goal of this experiment is to classify the sound in each non-silent period as either cockatoos, rainbow lorrikeets or others. For comparison purpose, we also input the recorded sound samples to the SVM classifier with MFCC features and the kNN classifier. The sound samples collected in this validation experiment result in 1,321 test vectors. The accuracy of our proposed ℓ_1-classifier is 70.17%, which is significantly better than that of SVM with MFCC at 41.03% and of kNN at 44.28%. This result is similar to those in Section 4.3 and demonstrate that proposed ℓ_1 is *robust to the environment noise, which makes it a good candidate for in-network classification tasks for ASNs deployed in remote harsh environment.*

6. RELATED WORK

6.1 Classification using Sparse Representation

The ℓ_1-classification method is first proposed in [38] for the purpose of face recognition. The method in [38] makes a number of assumptions, such as registration and scaling, which are only valid for images but do not apply to acoustic signals. There are a number of important differences between ℓ_1 classification for face recognition and acoustic signals. Section 3.1 has already discussed the alignment and dictionary redundancy problems. Another important difference is that face recognition is a single-label classification problem while acoustic classification is a multi-label one. This paper addresses the issues on making ℓ_1-classification possible for acoustic signals and solves the alignment, multi-label and dictionary redundancy problems.

Sparse approximation and ℓ_1-classification have also been applied to other acoustic classification problems. Sainath et al. [28] adopt the sparse approximation idea and use Approximate Bayesian Compressive Sensing (ABCS), which is essentially an ℓ_1 minimization technique, for phonetic classification. They show that in their setting, ABCS outperforms Gaussian Mixture Models (GMM), kNN and SVM methods, and offers an accuracy close to the best reported result in the literature. However, their approach still depends on feature selection and is only for single-label classification. In addition, they do not discuss whether their classification methods can be handled in real-time for real deployments.

6.2 Acoustic Classification using Machine Learning Techniques

Machine learning techniques have been heavily employed for animal acoustic classification and recognition (see for instance [19, 13, 22, 24]). Fagerlund [13] uses SVMs for bird species recognition. They use MFCC together with other descriptive parameters as the syllable candidates for feature extraction to classify the bird species. Huang et al. [24] adopt both kNN and SVMs for frog sound classification. They employ features like spectral centroid and signal bandwidth, which are well known in pattern recognition literature. Besides these, they propose a new feature called threshold-crossing rate to reduce the impact of noise in sound samples. However, how to select good features and kernel functions for SVMs are always difficult problems. Because of the difficulty of the feature selection and motivated by [38], we adopt the featureless spectrum domain samples for acoustic classification. Moreover, we investigate the system challenges of ℓ_1 classification methods in embedded ASNs in the realistic environment for remote acoustic sensing.

6.3 Classification on Wireless Sensor Networks

Although a network of embedded devices can largely increase the size of monitoring coverage area, it is challenging to realize classification methods on WSNs because of WSNs' limited computational ability and energy resource. In addition to the accuracy of the classification method, the computational speed and energy efficiency are also main concerns on WSNs. Recently, researchers have proposed classification methods for resource constrained environment according to these characteristics of WSNs. Sun [33] dynamically picks a part of the feature space rather than the entire one after feature selection to accelerate the classification. This method offers a good accuracy. However, the feature extraction procedure is very complicated. Gu et al. [18] design a hierarchical four-tier classification architecture. Each tier offers a classification based on the results from the lower ones, which enhances the accuracy. Hu et al. [22] design a hybrid ASN for monitoring amphibian population, which also chooses the strategy of in-network classification to save the transmission power consumption. Trifa et al. [36] implement hidden Markov models on the networked embedded devices for automated species recognition, where a wireless network plays a role in the acoustic sample recording and automatic real-time detection of the species. However, they rely on careful feature selection while we adopt a featureless approach. Duarte [12] applies local classification and global decision fusion strategy to classify the sound of moving vehicles in distributed sensor networks. The local classification can avoid the transmission loss, while the global decision can increase the accuracy. Su et al. [32] design a hierarchical aggregate classification protocol on WSNs. Each node only forwards its decision to the parent node to save energy. They also take the tradeoff between the energy consumption and the classification accuracy into consideration, and propose to use constrained hierarchical aggregate classifi-

cation to increase the accuracy at the cost of energy consumption. In contrast, our work aims to realize in-network acoustic classification without transmission of acoustic samples, which can cause high energy consumption because of high data rates of acoustic signals.

6.4 Compressive Sensing on Wireless Sensor Networks

Compressive sensing and sparse approximation can reduce the data dimension, which helps increase the computational efficiency and save the energy cost for transmission. The main question of applying compressive sensing to WSNs is how to use it to save energy in terms of minimizing the dimension of the transmitted data. Wu and Liu [39] use compressive sensing idea for soil moisture monitoring in WSNs for data compression. It results in a longer lifetime due to reduced amount of transmitted data. Misra et al. [26] use cross-correlation via sparse representation for range estimation in WSNs. The key idea there is to compress the signal samples using random projections and transmit them to a central device for reconstruction based on the knowledge of the sparsifying domain, which they call the correlation domain. Shen et al. [31] use compressive sensing as a dimension reduction tool to for background subtraction, reducing the amount of computations needed for calculating mixture of Gaussians (MoG) while retaining the accuracy. It is a fast background subtraction algorithm which can realize real-time tracking. Based on compressive sensing, Griffin and Tsakalides [17] study the reconstruction of audio signals using multiple sensor audio models, so as to detect and track a device that transmits periodical audio signals. The authors validate different sparsifying domains and reconstruction algorithms, and show that the system only requires the nodes to transmit part of the collected samples, which saves energy for transmission.

Different from these applications, we use compressive sensing and sparse approximation for classification purposes, which do not need accurate reconstruction. More importantly, in addition to random projections, we impose techniques such as silence removal and column reduction to further reduce the dimension of the problem. The resulting algorithm provides a better accuracy in noisy environment compared to the conventional machine learning methods without tedious feature selection.

7. CONCLUSION AND FUTURE WORK

In this paper, we propose a novel featureless, efficient and robust SAC framework for ASNs. We address a number of challenges to make SAC possible on resource constrained ASN nodes. In particular, in order to make the computationally expensive SAC method feasible for ASNs, we propose a column reduction algorithm to significantly reduce the size of the training dictionary. We also evaluate the performance of our proposed method using both simulations and experiments on an ASN testbed. The simulations are conducted with two real-life animal vocalization datasets. The results show that our proposed SAC method outperforms standard classifiers such as SVM and kNN for single-label classification, and ML-kNN for multi-label classification. The evaluations on the outdoor ASN testbed show that our column reduction method can significantly reduce the processing time so as to make real-time in-network classification using SAC method on embedded platforms possible. Our work can be extended in a few different directions. First, SAC on less powerful embedded platforms could be studied. Second, the effect of acoustic signals that overlap in both time and spectrum domains on classification is worth investigating. Third, the applications of SAC for military vehicles classification, indoor activity monitoring and speaker recognition are also interesting.

Acknowledgments. We thank our shepherd, Prof. Dr. Pedro José Marrón, and the anonymous reviewers for their helpful feedbacks on earlier versions of this paper.

8. REFERENCES

[1] A. O. Allen. *Probability, Statistics, and Queueing Theory with Computer Science Applications*. Academic Press, Inc., Orlando, FL, USA, 1978.

[2] E. Arráiz and O. Olivo. Competitive simulated annealing and tabu search algorithms for the max-cut problem. In *Proceedings of the 11th Annual conference on Genetic and evolutionary computation*, GECCO '09, pages 1797–1798, New York, NY, USA, 2009. ACM.

[3] R. Baraniuk, M. Davenport, R. DeVore, and W. M. A Simple Proof of the Restricted Isometry Property for Random Matrices. *Constr Approx*, 28:253–263, 2008.

[4] K. C. Barr and K. Asanović. Energy-aware lossless data compression. *ACM Trans. Comput. Syst.*, 24(3):250–291, Aug. 2006.

[5] T. Bokareva, W. Hu, S. Kanhere, B. Ristic, N. Gordon, T. Bessell, M. Rutten, and S. Jha. Wireless sensor networks for battlefield surveillance. In *Proceedings of the Land Warfare Conference (LWC)*, 2006.

[6] E. J. Candes, J. Romberg, and T. Tao. Robust uncertainty principles: exact signal reconstruction from highly incomplete frequency information. *IEEE Trans. on Inf. Theory*, 52(2):489–509, 2006.

[7] C.-C. Chang and C.-J. Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

[8] D. Donoho. Compressed sensing. *Information Theory, IEEE Transactions on*, 52(4):1289–1306, 2006.

[9] D. Donoho and X. Huo. Uncertainty principles and ideal atomic decomposition. *Information Theory, IEEE Transactions on*, 47(7):2845–2862, 2001.

[10] D. Donoho and Y. Tsaig. Fast Solution of ℓ_1 -Norm Minimization Problems When the Solution May Be Sparse. *Information Theory, IEEE Transactions on*, 54(11):4789–4812, 2008.

[11] D. L. Donoho and M. Elad. Optimally sparse representation in general (nonorthogonal) dictionaries via ℓ_1 minimization. *Proceedings of the National Academy of Sciences*, 100(5):2197–2202, 2003.

[12] M. Duarte. Vehicle classification in distributed sensor networks. *Journal of Parallel and Distributed Computing*, 2004.

[13] S. Fagerlund. Bird species recognition using support vector machines. *EURASIP J. Appl. Signal Process.*, 2007(1):64–64, 2007.

[14] D. Friedlander, C. Griffin, N. Jacobson, S. Phoha, and R. R. Brooks. Dynamic agent classification and tracking using an ad hoc mobile acoustic sensor network. *EURASIP J. Appl. Signal Process.*, 2003:371–377, Jan. 2003.

[15] L. Girod, M. Lukac, V. Trifa, and D. Estrin. The design and implementation of a self-calibrating distributed acoustic sensing platform. In *SenSys*, pages 71–84. ACM, 2006.

[16] F. Glover and M. Laguna. *Tabu Search*. Kluwer Academic Publishers, Norwell, MA, USA, 1997.

[17] A. Griffin and P. Tsakalides. Compressed sensing of audio signals using multiple sensors. In *Proceedings of 16th European Signal Processing Conference*, EUSIPCO 2008, 2008.

[18] L. Gu, D. Jia, P. Vicaire, T. Yan, L. Luo, A. Tirumala, Q. Cao, T. He, J. A. Stankovic, T. Abdelzaher, and B. H. Krogh. Lightweight detection and classification for wireless sensor networks in realistic environments. In *Proceedings of the 3rd international conference on Embedded networked sensor systems*, SenSys '05, pages 205–217, New York, NY, USA, 2005. ACM.

[19] G. Guo and S. Li. Content-based audio classification and retrieval by support vector machines. *Neural Networks, IEEE Transactions on*, 14(1):209–215, 2003.

[20] Y. Guo and M. Hazas. Localising speech, footsteps and other sounds using resource-constrained devices. In *Information Processing in Sensor Networks (IPSN), 2011 10th International Conference on*, pages 330 –341, april 2011.

[21] Y. Hao, B. Campana, and E. Keogh. Monitoring and mining animal sounds in visual space. *Journal of Insect Behavior*, pages 1–28, 2012.

[22] W. Hu, N. Bulusu, C. T. Chou, S. Jha, A. Taylor, and V. N. Tran. Design and evaluation of a hybrid sensor network for cane toad monitoring. *ACM Trans. Sen. Netw.*, 5(1):4:1–4:28, 2009.

[23] W. Hu, C. T. Chou, S. Jha, and N. Bulusu. Deploying long-lived and cost-effective hybrid sensor networks. *Ad Hoc Networks*, 4(6):749 – 767, 2006.

[24] C.-J. Huang, Y.-J. Yang, D.-X. Yang, and Y.-J. Chen. Frog classification using machine learning techniques. *Expert Syst. Appl.*, 36(2):3737–3743, 2009.

[25] A. Krause and C. Guestrin. Optimizing Sensing: From Water to the Web. *Computer*, 42:38–45, 2009.

[26] P. Misra, W. Hu, M. Yang, and S. Jha. Efficient cross-correlation via sparse representation in sensor networks. In *Proceedings of the 11th international conference on Information Processing in Sensor Networks*, IPSN '12, pages 13–24, New York, NY, USA, 2012. ACM.

[27] J. Renaud, G. Laporte, and F. F. Boctor. A tabu search heuristic for the multi-depot vehicle routing problem. *Computers & Operations Research*, 23(3):229 – 235, 1996.

[28] T. Sainath, A. Carmi, D. Kanevsky, and B. Ramabhadran. Bayesian compressive sensing for phonetic classification. In *Acoustics Speech and Signal Processing (ICASSP), 2010 IEEE International Conference on*, pages 4370–4373, 2010.

[29] M. Salman Asif and J. Romberg. Dynamic updating for ℓ_1 minimization. *Selected Topics in Signal Processing, IEEE Journal of*, 4(2):421 –434, april 2010.

[30] R. E. Schapire and Y. Singer. BoosTexter: A Boosting-based System for Text Categorization. *Machine Learning*, 39(2/3):135–168, 2000.

[31] Y. Shen, W. Hu, J. Liu, M. Yang, B. Wei, and C. T. Chou. Efficient background subtraction for real-time tracking in embedded camera networks. In *Proceedings of the 10th ACM Conference on Embedded Networked Sensor Systems*, SenSys '12, New York, NY, USA, 2012. ACM.

[32] L. Su, J. Gao, Y. Yang, T. F. Abdelzaher, B. Ding, and J. Han. Hierarchical aggregate classification with limited supervision for data reduction in wireless sensor networks. In *Proceedings of the 9th ACM Conference on Embedded Networked Sensor Systems*, SenSys '11, pages 40–53, New York, NY, USA, 2011. ACM.

[33] Y. Sun. Dynamic target classification in wireless sensor networks. *Pattern Recognition*, 2008.

[34] A. Taylor, G. Watson, G. Grigg, and H. McCallum. Monitoring frog communities: an application of machine learning. In *Proceedings of the eighth annual conference on Innovative applications of artificial intelligence*, pages 1564–1569. AAAI Press, 1996.

[35] Z. C. Taysi, M. A. Guvensan, and T. Melodia. Tinyears: spying on house appliances with audio sensor nodes. In *Proceedings of the 2nd ACM Workshop on Embedded Sensing Systems for Energy-Efficiency in Building*, BuildSys '10, pages 31–36, New York, NY, USA, 2010. ACM.

[36] V. Trifa, A. Kirschel, C. E. Taylor, and E. E. Vallejo. Automated species recognition of antbirds in a mexican rainforest using hidden markov models. *Journal of the Acoustical Society of America*, 123(4):2424–2431, April 2008.

[37] G. Vaca-Castaño and D. Rodriguez. Using syllabic mel cepstrum features and k-nearest neighbors to identify anurans and birds species. In *Signal Processing Systems (SIPS), 2010 IEEE Workshop on*, pages 466–471, oct. 2010.

[38] J. Wright, A. Yang, A. Ganesh, S. Sastry, and Y. Ma. Robust face recognition via sparse representation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 31(2):210–227, 2009.

[39] X. Wu and M. Liu. In-situ soil moisture sensing: measurement scheduling and estimation using compressive sensing. In *Proceedings of the 11th international conference on Information Processing in Sensor Networks*, IPSN '12, pages 1–12, New York, NY, USA, 2012. ACM.

[40] Y. Yang, L. Wang, D. K. Noh, H. K. Le, and T. F. Abdelzaher. Solarstore: enhancing data reliability in solar-powered storage-centric sensor networks. In *Proceedings of the 7th international conference on Mobile systems, applications, and services*, MobiSys '09, pages 333–346, New York, NY, USA, 2009. ACM.

[41] M.-L. Zhang and Z.-H. Zhou. Ml-knn: A lazy learning approach to multi-label learning. *Pattern Recogn.*, 40(7):2038–2048, July 2007.