




Article

Animal Sound Classification Using Dissimilarity Spaces

Loris Nanni ^{1,*} , Sheryl Brahnam ² , Alessandra Lumini ³  and Gianluca Maguolo ¹

¹ Department of Information Engineering, University of Padova, Via Gradenigo 6, 35131 Padova, Italy; gianluca.maguolo@phd.unipd.it

² Department of Information Technology and Cybersecurity, Missouri State University, 901 S, National Street, Springfield, MO 65804, USA; sbrahnam@missouristate.edu

³ Department of Computer Science and Engineering, University of Bologna, Via dell'Università 50, 47521 Cesena, Italy; alessandra.lumini@unibo.it

* Correspondence: loris.nanni@unipd.it

Received: 30 October 2020; Accepted: 26 November 2020; Published: 30 November 2020



Abstract: The classifier system proposed in this work combines the dissimilarity spaces produced by a set of Siamese neural networks (SNNs) designed using four different backbones with different clustering techniques for training SVMs for automated animal audio classification. The system is evaluated on two animal audio datasets: one for cat and another for bird vocalizations. The proposed approach uses clustering methods to determine a set of centroids (in both a supervised and unsupervised fashion) from the spectrograms in the dataset. Such centroids are exploited to generate the dissimilarity space through the Siamese networks. In addition to feeding the SNNs with spectrograms, experiments process the spectrograms using the heterogeneous auto-similarities of characteristics. Once the similarity spaces are computed, each pattern is “projected” into the space to obtain a vector space representation; this descriptor is then coupled to a support vector machine (SVM) to classify a spectrogram by its dissimilarity vector. Results demonstrate that the proposed approach performs competitively (without ad-hoc optimization of the clustering methods) on both animal vocalization datasets. To further demonstrate the power of the proposed system, the best standalone approach is also evaluated on the challenging Dataset for Environmental Sound Classification (ESC50) dataset.

Keywords: audio sound classification; clustering; prototype selection; Siamese network; dissimilarity space

1. Introduction

Over the past decade, research in sound classification and recognition has gained in popularity and rapidly broaden in its application from the more traditional focus on speech recognition [1] and music genre classification [2] to biometric identification [3], computer-aided heart sound detection [4], environmental audio scene and sound recognition [5,6], biodiversity assessment [7], human voice classification and emotion recognition [8], English accent classification, and gender identification [9], to list a few of a wide range of application areas. As with research in pattern recognition generally, the features fed into classifiers were initially engineered, which, in the case of sound applications, meant extracting from raw audio traces such descriptors as the Statistical Spectrum Descriptor and Rhythm Histogram [10].

In the past decade, researchers began exploring the possibility of visually representing audio signals to apply more powerful image classification descriptors and techniques. Initially, visual representations of audio traces centered around the display overtime of the frequency spectrum: examples of these representations include the spectrogram [11] and Mel-frequency Cepstral Coefficients spectrogram [12].

Although a spectrogram is typically a graph with two dimensions (time and frequency), additional dimensions can be included, such as pixel intensity [13], which includes at each time step the representation of an audio signal's amplitude in a specific frequency. Initially, popular texture descriptors, such as Haralick's Grey Level Co-occurrence Matrices (GLCMs) [14], Gabor filters [15], and Local Binary Patterns (LBP) [16] and its variants [17] were extracted from these spectrograms for the task of music genre classification [18–20]. Fusions of a large set of state-of-the-art texture descriptors were then experimentally applied to spectrograms, and certain combinations were shown to enhance further the accuracy of music genre classification [2].

The rise in popularity of deep learning due to affordable graphic processing units (GPUs) changed the trajectory in machine learning research [21]. Deep learners, such as the convolutional neural network (CNN), produced far superior results to most other classifiers when it came to image classification [22]. Consequently, more attention was placed on representing acoustic traces through visual representations so that deep learning approaches could be applied. Engineered features diminished in importance since deep classifiers learn which patterns perform best for a specific problem during the training process. However, engineered features have been shown to augment deep learning approaches when fused. Early work with CNNs applied to visual representations obtained state-of-the-art in chord detection and recognition [23,24] and in music genre classification [25]. In [25], for example, spectrograms were converted into GCLM maps and trained on CNNs. In [26], canonical approaches, such as LBP-trained SVMs were fused with CNNs and shown to outperform previous systems.

Another development in sound classification involves the design of deep learners and feature sets specific to audio classification. For example, in [27], the authors explored variations in CNN architecture and parameters; in [28] a novel *sparse coding* CNN was developed and shown to perform well if not better than the state-of-the-art for sound event recognition and retrieval. Also of note is the hybrid multimodal deep learning approach proposed in [29] for multilabel music genre classification that combined album cover images, reviews, and audio tracks; this system was shown to outperform the single-modality approaches. In [30] the authors present a gated CNN and a temporary attention-based classification system for audio classification from weakly labeled data.

When it comes to animal sound classification, the focus of this study, fusions of CNNs with other methods to classify animals have been evaluated in [31] and [32] for fish identification using the Fish and MBARI benthic animal dataset and in [33] for bioacoustic bird species classification using a dataset of audio samples from 43 species. In [31], the authors combined engineered features with CNNs, and in [33], deep learning was combined with shallow learning. Both demonstrated that the fusions performed better than the standalone approaches.

Animal sound classification is a vibrant area of research. Many benchmark datasets are available for a variety of animals, such as bats [34], birds [34,35], whales [36], cats [37], and frogs [35]. In the literature, animal audio classification is broadly divided into two categories: the CNN approach discussed above and fingerprinting [38], which involves a compact audio representation for comparing audio segments in terms of similarity and dissimilarity [39]. Both approaches, however, suffer from limitations: fingerprinting can only find exact matches, and CNNs require large datasets for accurate training; many animal datasets are small because of the difficulty of collecting samples.

The superiority of combining the deep learning approach with fingerprinting is demonstrated in [40], where a Siamese Neural Network (SNN) produced semantic descriptions of audio signals. SNNs have been applied to sound classification in [40–42] and have the advantage over the canonical CNN in their ability to generalize. In [43], a system was developed based on dissimilarity spaces, such as that proposed in [44] for brain image classification, where a distance model was learned by training a SNN [45] on dissimilarity values. This system combined several clustering approaches to obtain a dissimilarity space; each pattern was represented in such a space, and the resulting descriptors were used to train a general purpose classifier (SVM). The clustering methods transformed the spectrograms in a bird [46] and cat [37,47] dataset to a set of centroids that were used to generate a vector space

representation for each pattern. This vector was then used to train an SVM. Results showed that this approach worked better than the standalone CNNs.

The system proposed in this work is similar to [43] in that it generates a dissimilarity space from the training set using an SNN to define a distance function from the input spectrograms. The objective at this point in the process is to maximize the distance separating the patterns of the different classes. Unlike [43], however, four different CNN architectures are selected for the twin classifiers, and both the original input spectrograms and the spectrograms processed by Heterogeneous Auto-Similarities of Characteristics [48] make up the inputs to the SNNs. In the testing phase, the unknown pattern is compared to the centroids of the dissimilarity space using the different SNNs for comparing two samples and measure their dissimilarity. Although it is the case that the entire training set can function as the centroids of the dissimilarity space, it is desirable to reduce dimensionality by selecting a smaller number of prototypes. In this work, both a supervised and unsupervised clustering algorithm are used to reduce dimensionality. The dissimilarity space represents each input (both the original and the processed spectrograms) by its distance from each of the centroids, or prototypes, a distance that is learned by the SNNs. In other words, the SNNs compare a given spectrogram to each of the centroids to obtain a dissimilarity feature vector. A support vector machine (SVM) is then trained on these features.

The approach taken in this paper is evaluated on the same animal vocalization datasets as in [43], i.e., on cats [37] and birds [7]. In addition, the system is tested on the challenging benchmark dataset for Environmental Sound Classification (ESC-50) [49]. As in [43], the most effective system is the result of a fusion of classifiers, i.e., the sum rule of SVMs trained on different dissimilarity spaces generated by changing the value of k in the clustering approaches and network topologies. Performance is compared with both the state-of-the-art as well as with fusions with the state-of-the-art. Results demonstrate the power of using dissimilarity spaces based on an ensemble of SNNs, particularly when coupled with a standard CNN approach. The MATLAB code of the proposed method is freely available at <https://github.com/LorisNanni>.

The remainder of this paper is broken down into the following sections. In Section 2, a detailed overview of the proposed approach is provided; in Section 3, SNN is discussed along with the different CNN architectures that make up the subnetworks. In Section 4, the supervised and unsupervised clustering methods are outlined. In Section 5, experimental results are presented along with comparisons with the state-of-the-art. The paper concludes in Section 6 with an overview and some suggestions for future research.

2. Proposed System

The system presented here for spectrogram classification extends that proposed in [43] and is schematized in Figure 1, which illustrates the approach using one SNN. The pseudocode in Algorithms 1 and 2 that correspond to Figure 1 are detailed in the remainder of this section.

The process begins by generating a similarity space during the training phase via a learning distance measure $d(x, y)$ from a set of prototypes $P = p_1, \dots, p_k$. The distance measure is learned by four SNNs trained (1) to maximize the similarity between pairs of spectrograms belonging to the same class and (2) to minimize the similarity for pairs of spectrograms belonging to different classes. The set of prototypes generated in this phase are the k centroids of the clusters produced by a clustering approach. What results is a feature vector $f \in R^k$ that represents training sample x in the dissimilarity space, where a given f_i is the distance between x and the prototype p_i : $f_i = d(x, p_i)$. Once these features have been calculated, they are used to train an SVM classifier.

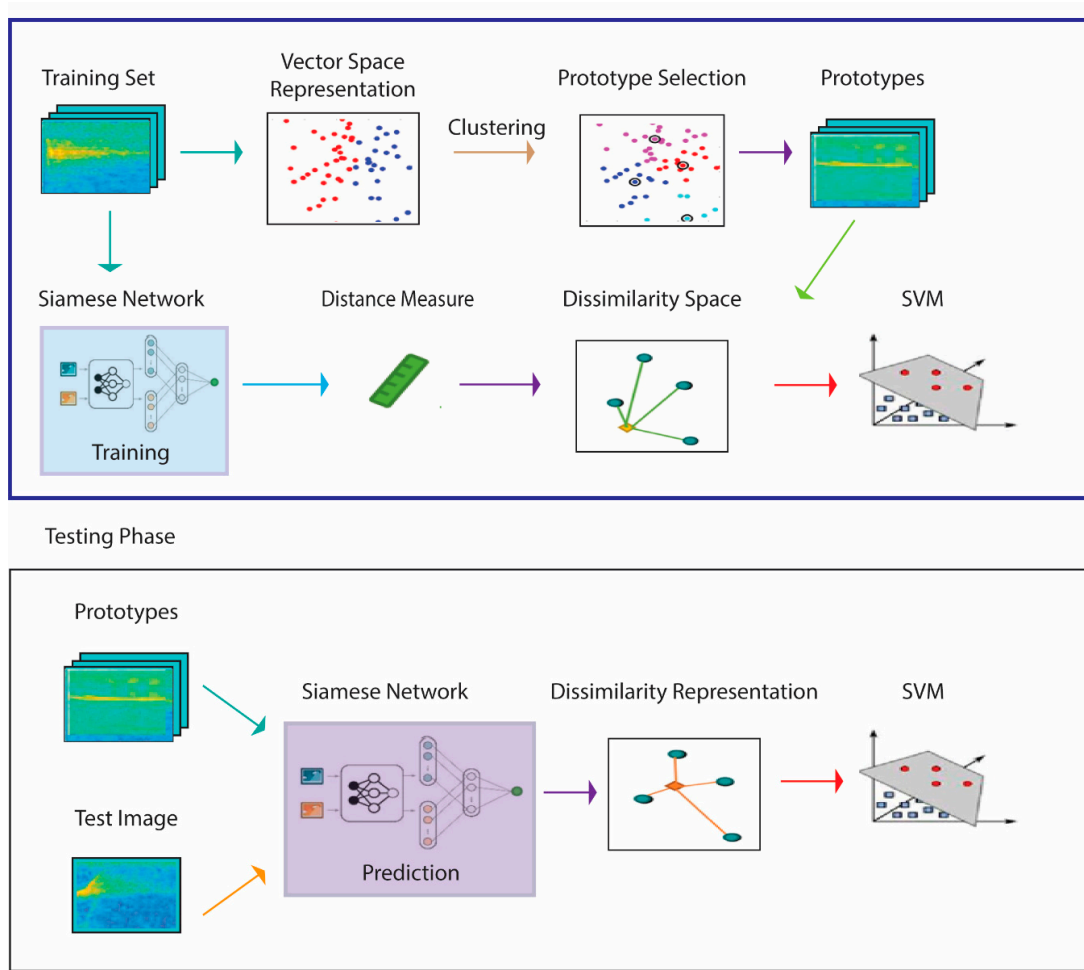


Figure 1. Generalized schematic of the proposed approach.

In the testing phase, each unknown pattern is represented by its projection in a dissimilarity space: the descriptor is obtained by calculating the pattern distance to the set of prototypes P , and the resulting feature vectors of the input images are then classified by SVM. In our experiments, we test both spectrograms and HASC [48], a 2D descriptor extracted from the spectrogram, as the input of the classification process. The method for extracting HASC is outlined in Section 2.5.

Algorithm 1. Training phase

Input: Training images ($imgsTrain$), training labels ($labelTrain$), number of training iterations ($trainIterations$), batch size ($trainBatchSize$), number of centroids (k), and clustering technique ($type$).

Output: Trained SNN ($tSNN$), set of centroids (C), and trained SVM (svm).

1: $tSNN \leftarrow \text{TRAINSIAMESE}(imgsTrain, labelTrain, trainIterations, trainBatchSize)$

2: $p \leftarrow \text{CLUSTERING}(imgsTrain, labelTrain, k, type)$

3: $F \leftarrow \text{GETDISSPACEPROJECTION}(imgsTrain, P, tSNN)$

4: $tSVM \leftarrow \text{TRAINSVM}(labelTrain, F)$

Algorithm 2. Testing phase

Input: Test images ($imgsTest$), trained SNN ($tSNN$), Set of centroids (C), Trained SVM ($tSVM$).

Output: Actual test labels ($labelTest$).

1: $F \leftarrow \text{GETDISSPACEPROJECTION}(imgsTest, P, tSNN)$

2: $labelTest \leftarrow \text{PREDICTSVM}(F, tSVM)$

2.1. Siamese Neural Network Training

To define a similarity measure inside the Dissimilarity Space, a Siamese Network is trained to compare a couple of spectrograms and return a similarity value that is the greater value if they belong to the same class and the lower value if they belong to different classes. The pseudocode for training an SNN, which is called in step 1 of Algorithm 1, is reported in Algorithm 3. The SNN architecture is defined in steps 2 and 3 (more details are given in Section 3). The training process consists in repeating Steps 5–8: random extraction of *batchSize* spectrogram pairs from the training set (via the function GETBATCH), computation of loss and gradients for each pair passed to the SNN, and upgrade of the SNN accordingly (i.e., the subnet and the fully connected (FN) layer of the Siamese network).

Algorithm 3. Siamese training pseudocode

Input: Training image (*trainImgs*), training labels (*trainLabels*), batch size (*batchSize*), and iterations (*numberOfIterations*).

Output: Trained SNN (*tSNN*).

```

1: function TRAINSVM
2:   subnet ← NETWORK([inputLayer, ..., FullyConnectedLayer])
3:   fcWeights ← randomWeights
4:   for iteration from 1 to numberOfIterations do
5:     X1, X2, pairLabels ← GETBATCH (trainImgs, trainLabels, batchSize)
6:     gradients, loss ← EVALUATE(subnet, X1, X2, pairLabels)
7:     UPDATE(subnet, gradients)
8:     UPDATE(fcWeights, gradients)
9:   end for
10:  return tSNN ← subnet, fcWeights
11: end function

```

Note: if SNN fails to converge on the training set, the training phase is repeated.

2.2. Prototype Selection

Prototype selection involves extracting a total of k prototypes from the training set in order to reduce the dimensionality of the dissimilarity space, which would be too high to maintain each training sample as a prototype. Dimensionality can be reduced by employing clustering techniques to calculate k centroids. In this work, we perform both prototype selection separately for each class (k_c prototypes per class) and via global (i.e., unsupervised) clustering (k prototypes): in both cases, results are compared considering the same final dimension. Algorithm 4 presents the pseudocode for prototype selection. As can be observed, a clustering technique is selected from a set of four possible clustering methods, each of which is employed separately to cluster the training samples. For the sake of space, only the pseudocode for the supervised clustering methods is included, though it should be noted that this work used both supervised and unsupervised clustering approaches.

2.3. Projection in the Dissimilarity Space

Classically, classifiers are trained to predict patterns within a feature space. It is also possible, as demonstrated here, for patterns to be projected in a dissimilarity space such that each pattern x is characterized by its dissimilarity to a set of prototypes $P = p_1, \dots, p_k$ and by a dissimilarity vector defined as

$$F(x) = [d(x, p_1), \dots, d(x, p_i), \dots, d(x, p_k)], \quad (1)$$

where the similarity of pattern $d(x, y)$ is obtained using a trained SNN.

Algorithm 4. Clustering pseudocode

Input: Training images (*imgsTrain*), training labels (*labelTrain*), number of clusters (*k*), and clustering technique (*type*).

Output: Centroids *P*.

```

1: function CLUSTERING
2:   numClasses ← number of classes from labelTrain
3:   kc ← k/numClasses
4:   for i from 1 to numClasses do
5:     images ← images of the class i from imgsTrain
6:     switch type do
7:       case "k-means" Pi ← KMEANS(imgs,kc)
8:       case "k-medoids" Pi ← KMEDOIDS(imgs,kc)
9:       case "hierarchical" Pi ← HIERARCHICAL(imgs,kc)
10:      case "spectral" Pi ← SPECTRAL(imgs,kc)
11:    P ← P ∪ Pi
12:   end for
13:   return P
14: end function

```

The projection of a set of images (from the training or the testing set) into a dissimilarity space \mathcal{R}^k is described in Algorithm 5. Each image of the input set (stored in the variable *X*, see step 3) is compared with the *k* prototypes (stored in *P*) according to the dissimilarity measure learned by the trained SNN (PREDICTSIAMESE function, see step 4). The number of centroids is a parameter, and different values of *k* were tested (depending on the number of classes *c*): $k = kc * c$, $kc = \{15, 30, 45, 60\}$. The output is the feature space *F* that includes the projections of all the input images *imgs*.

Algorithm 5. Projection in the Dissimilarity space pseudocode

Input: Images (*imgs*), Centroids (*P*), number of centroids (*k*), and trained SNN (*tSNN*).

Output: Feature vectors (*F*).

```

1: function GETDISSPACEPROJECTION
2:   for j from 1 to SIZE(imgs) do
3:     X ← imgs[j]
4:     F[j] ← PREDICTSIAMESE(tSNN, X, P)
5:   end for
6:   return F
7: end function

```

2.4. Classification by SVM

SVM [50] is a well-known binary learner that represents training samples as points in space. The goal of SVM training (function TRAINSVM) is to find at least one hyperplane such that it separates the data that belongs to each of the two classes. Prediction (function PREDICTSVM) is accomplished by mapping an unseen pattern to the side of the hyperplane representing the class for a given data point.

SVM, as defined above, does a poor job discriminating input that is not linearly separable in its original space. This difficulty can be overcome by selecting kernel functions that map the data into a higher-dimensional space where separation is possible. In this work, we chose a radial basis function kernel to map a single vector to a vector of higher dimensionality.

Although SVM is binary, it can be applied to nonbinary or multilabel problems by training an ensemble of SVMs and combining their decisions. In the experiments reported here, the One-Against-All method is used, where an SVM is trained systematically to distinguish each class against all the others combined. The pattern is then predicted to belong to that class that produces the highest confidence score.

2.5. Heterogeneous Auto-Similarities of Characteristics (HASC)

HASC [48] is applied to heterogeneous dense feature maps. It encodes linear relations by covariances (COV) and nonlinear associations with entropy combined with mutual information (EMI). Three reasons for considering covariance matrices as descriptors are as follows: (1) they are low in dimensionality, (2) robust to noise (but with the exception that outlier pixels can render them more sensitive to noise), and (3) the covariance among two features is optimally able to encapsulate the features of the joint PDF (but with the caveat that they be linked by a linear relation). HASC obviates these limitations by combining COV with EMI.

The entropy (E) of a random variable measures the uncertainty of its value, and the mutual information (MI) of two random variables captures their generic linear and nonlinear dependencies. The way HASC utilizes these advantages is by dividing an image into patches from which it generates an EMI matrix ($d \times d$), such that the main diagonal entries encapsulate the amount of unpredictability of the d features. The off-diagonal entries (element i, j) capture the mutual dependency between two features, that is, the i -th and j -th feature. HASC is the concatenation of vectorized EMI and COV. Specifically, the MI of a pair of random variables A, B is

$$MI(A, B) = \int_A \int_B p(a, b) \log \left(\frac{p(a, b)}{p(a)p(b)} \right) db da, \quad (2)$$

where $p(a)$, $p(b)$, and $p(a, b)$ are the PDF of A , the PDF of B , and their joint PDF, respectively. If $A = B$, then MI is the entropy of A :

$$E(A) = MI(A, A) = - \int_A p(a) \log(p(a)) da. \quad (3)$$

If there exists a finite set K of realization pairs, MI can be estimated as a sample mean inside the logarithm, thus:

$$MI(A, B) \approx \frac{1}{K} \sum_{k=1}^K \log \left(\frac{p(a_k, b_k)}{p(a_k)p(b_k)} \right). \quad (4)$$

A fast and efficient method for calculating from the K realizations the probabilities inside the logarithm is to estimate them by building a joint 2D normalized histogram of values A and B , such that each $p(a_k, b_k)$ is estimated taking the value of the 2D histogram bin containing the pair a_k, b_k . In this way, $p(a_k)$ and $p(b_k)$ can be estimated by summing all the bins corresponding to a_k and b_k , respectively. Thus, the i, j -th component of the EMI matrix related to the patch P can be defined as

$$EMI_{p\{ij\}} = \frac{1}{K} \sum_{k=1}^K \log \left(\frac{\tilde{p}(z_{ki}, z_{kj})}{\tilde{p}(z_{ki})\tilde{p}(z_{kj})} \right), \quad (5)$$

where $\tilde{p}(\dots)$ and $\tilde{p}(\cdot)$ are the probabilities estimated with the histogram and z_{ki} is the i -th feature at pixel K .

In this study, HASC is extracted from the whole spectrogram. Given the function HASC, the output FEAT is a three-dimensional matrix ($w \times h \times d$) containing the features extracted from the image (d is the number of low-level features). The number of bins used to evaluate the histograms in the EMI computation is 28, and the number of low-level features is 6 (default parameters). FEAT is reshaped to build the vector $img = [\text{FEAT}(:, :, 1) \text{ FEAT}(:, :, 2); \text{FEAT}(:, :, 3) \text{ FEAT}(:, :, 4); \text{FEAT}(:, :, 5) \text{ FEAT}(:, :, 6)]$, and img is resized to the right dimension for the input into a CNN.

3. Siamese Neural Network (SNN)

SNN, initially proposed in [45], is a class of neural networks that contains two or more twin networks that share the same weights and parameters. As illustrated in Figure 2, an SNN has two

inputs that compare two patterns and one output that corresponds to the similarity between the two inputs. In other words, an SNN identifies correlations between two different input patterns (see [51] for a fairly comprehensive overview of SNN). As shown in Figure 2, the SNNs used in this work are composed of five components, as described below.

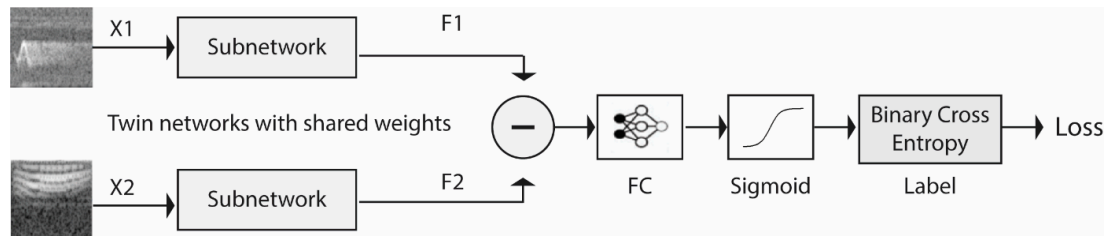


Figure 2. Siamese neural network architecture.

3.1. The Two Identical Twin Subnetworks

In this study, four twin CNN subnetworks are utilized. CNNs are constructed by assembling specialized layers composed of neurons. Some of the more common layers found in CNN architectures include convolutional, activation (ReLU), pooling, and fully connected (FC) layers. The specific CNN architecture for the four twin subnetworks are outlined in Table 1.

Table 1. Convolutional neural network (CNN) Siamese networks (1–4) layers.

Siamese Network 1				
Layers	Activations	Learnable	Filter Size	Num. of Filters
Input Layer	224×224			
2D Convolution	$215 \times 215 \times 64$	6464	10×10	64
ReLU	$215 \times 215 \times 64$	0		
Max Pooling	$107 \times 107 \times 64$	0	2×2	
2D Convolution	$101 \times 101 \times 128$	401,536	7×7	128
ReLU	$101 \times 101 \times 128$	0		
Max Pooling	$50 \times 50 \times 128$	0	2×2	
2D Convolution	$47 \times 47 \times 128$	262,272	4×4	128
ReLU	$47 \times 47 \times 128$	0		
Max Pooling	$23 \times 23 \times 128$	0	2×2	
2D Convolution	$19 \times 19 \times 64$	204,864	5×5	64
ReLU	$19 \times 19 \times 64$	0		
Fully Connected	4096	94,638,080		
Siamese Network 2				
Layers	Activations	Learnable	Filter Size	Num. of Filters
Input Layer	224×224	0		
2D Convolution	$220 \times 220 \times 64$	1664	5×5	64
LeakyReLU	$220 \times 220 \times 64$	0		
2D Convolution	$216 \times 216 \times 64$	102,464	5×5	64
LeakyReLU	$216 \times 216 \times 64$	0		
Max Pooling	$108 \times 108 \times 64$	0	2×2	
2D Convolution	$106 \times 106 \times 128$	73,856	3×3	128
LeakyReLU	$106 \times 106 \times 128$	0		
2D Convolution	$104 \times 104 \times 128$	147,584	3×3	128
LeakyReLU	$104 \times 104 \times 128$	0		
Max Pooling	$52 \times 52 \times 128$	0	2×2	
2D Convolution	$49 \times 49 \times 128$	262,272	4×4	128
LeakyReLU	$49 \times 49 \times 128$	0		
Max Pooling	$24 \times 24 \times 128$	0	2×2	
2D Convolution	$20 \times 20 \times 64$	204,864	5×5	64
LeakyReLU	$20 \times 20 \times 64$	0	5×5	
Fully Connected	2048	52,430,848		

Table 1. Cont.

Siamese Network 3				
Layers	Activations	Learnable	Filter Size	Num. Filters
Input Layer	224 × 224			
2D Convolution	55 × 55 × 128	6400	7 × 7	128
Max Pooling	27 × 27 × 128	0	2 × 2	
2D Convolution	23 × 23 × 256	819,456	5 × 5	256
ReLU	23 × 23 × 256	0		
2D Convolution	19 × 19 × 128	819,328	5 × 5	128
Max Pooling	9 × 9 × 128	0	2 × 2	
2D Convolution	7 × 7 × 64	73,792	3 × 3	64
ReLU	7 × 7 × 64	0		
Max Pooling	3 × 3 × 64	0	2 × 2	
Fully Connected	4096	2,363,392		
Siamese Network 4				
Layers	Activations	Learnable	Filter Size	Num. of Filters
Input Layer	224 × 224			
2D Convolution	218 × 218 × 128	6400	7 × 7	128
Max Pooling	54 × 54 × 128	0	4 × 4	
ReLU	54 × 54 × 128	0		
2D Convolution	50 × 50 × 256	819,456	5 × 5	256
ReLU	50 × 50 × 256	0		
2D Convolution	48 × 48 × 64	147,520	3 × 3	64
Max Pooling	24 × 24 × 64	0	2 × 2	
2D Convolution	22 × 22 × 128	73,856	3 × 3	128
ReLU	22 × 22 × 128	0		
2D Convolution	18 × 18 × 64	204,864	5 × 5	64
Fully Connected	4096	84,938,752		

Two activation functions are explored here: the ReLU activation function [52] and the Leaky ReLU [53]. The well-known ReLU activation function for the points (x_i, y_i) is defined as:

$$y_i = f(x_i) = \begin{cases} 0, & x_i < 0 \\ x_i, & x_i \geq 0 \end{cases} \quad (6)$$

The Leaky ReLU variation of ReLU is defined as

$$y_i = f(x_i) = \begin{cases} ax_i, & x_i < 0 \\ x_i, & x_i \geq 0 \end{cases} \quad (7)$$

The subnetworks of the Siamese CNNs 1–4 each learn the features that best represent the information in the spectrograms that are the input patterns to the two input nodes (X1 and X2) and return either a 2048 or a 4096-dimensional feature vector (F1 and F2). The subnetworks share the same parameters and training weights.

3.2. Subtract Block, FC Layer, and Sigmoid Function

The Subtract Block calculates the feature vector Y as the absolute value of the difference between the resulting vectors of the two subnetworks:

$$Y = |F1 - F2|. \quad (8)$$

Following the method outlined in [44], the FC layer learns the distance model for calculating the dissimilarity. Y , the result of the Subtract Block, is processed by the FC block to produce the measure of dissimilarity between two spectrogram patterns. Because of this block, the resulting dissimilarity measure is not a metric since it does not have the following properties: identity and triangle inequality. It does, however, have the properties of symmetry and compactness as defined in [22], so that a

sufficiently small perturbation of an object leads to an arbitrary small dissimilarity value between the disturbed object and its original.

The sigmoid function is then applied to the dissimilarity value to transform it to a probability value in the range $[0,1]$:

$$S(x) = 1/(1 + e^{-x}). \quad (9)$$

4. Clustering

Clustering is a procedure that divides unlabeled patterns into groups that maximize the commonality between members within a group and their differences with members belonging to other groups. Clustering techniques often calculate the mean vector, or centroid, of all the patterns within a cluster when forming the clusters. Centroids encapsulate salient characteristics of patterns belonging to a cluster; for this reason, they can be used to reduce the dissimilarity space size without losing too much important information. Even more information can be retained if the number of centroids representing each class is increased.

Both supervised and unsupervised clustering approaches are considered here. If kc clusters are extracted with a supervised approach, then, in each of NK classes, $kc \times NK$ clusters are extracted using unsupervised clustering on the training set. A description of the four clustering methods used in this study follows.

4.1. K-Means

K-means is one of the most popular clustering algorithms. It partitions patterns into k clusters by placing each observation into a cluster based on the nearest centroid (according to the Euclidean Distance measure).

The standard k-means algorithm involves four steps:

1. Randomly select a set of centroids from among the data points.
2. For each data point x remaining in the training set, compute the distance $d(x)$ between it and the nearest centroid.
3. Recalculate new centroids via a weighted probability distribution.
4. Repeat Steps 2 and 3 until convergence.

4.2. K-Medoids

K-medoids is a clustering technique that follows the same general logic behind k-means but differs in the specific way it partitions data points into clusters: K-medoids minimizes the sum of distances between a given pattern and the center of that pattern's cluster. In short, the center of a cluster in K-Means ends up being the centroid of the cluster, but the center in K-Medoids is a member (medoid) of the cluster. In other words, a medoid is that member in a cluster whose sum of distances from all other members is minimal.

The standard K-medoids algorithm involves three steps:

1. Step one is a build-step where each k cluster is associated with a potential medoid. There are many ways to select the first medoid; the standard MATLAB's implementation does this employing the k-means++ heuristic.
2. Step two is a swap-step where each point in a cluster is tested as a potential medoid by checking whether the sum of the within-cluster distances is smaller when using that point as the medoid. Every point is then assigned to the cluster with the closest medoid.
3. The last step repeats previous steps until convergence.

4.3. Spectral

Spectral clustering partitions data into clusters starting from the adjacency matrix representing the undirected similarity graph of the patterns. Each pattern is a node of the graph, and two nodes are connected only if their similarity is larger than a threshold (typically set to 0).

This clustering algorithm involves the following matrices:

- The similarity matrix M , whose cell m_{ij} is the similarity value of two patterns (i.e., two spectrograms s_i, s_j);
- The degree matrix D , which is a diagonal matrix that is obtained by summing the rows of M :

$$D_g(i, i) = \sum_j m_{i,j}; \quad (10)$$

- The Laplacian matrix L , which is defined as

$$L = D_g - M. \quad (11)$$

The algorithm for spectral clustering is a five-step process:

1. Define a local neighborhood for each data point in the dataset (there are many ways to define a neighborhood; the nearest-neighbor method is the default setting in the MATLAB implementation of spectral clustering). Then compute the local similarity matrix of each pattern in the neighborhood.
2. Calculate the Laplacian matrix L .
3. Create a matrix V containing columns v_1, \dots, v_k , where the columns are the k eigenvectors, i.e., the *spectrums* (hence the name), corresponding to the k smallest eigenvalues of L .
4. Perform k-means or k-medoids clustering by treating each row of V as a datapoint.
5. Cluster the original pattern according to the assignments of their corresponding rows.

4.4. Hierarchical Clustering

Hierarchical clustering partitions data by creating a tree of clusters divided into n levels selected for the specific classification task. In general, hierarchical clustering is divided into two types:

1. Agglomerative, where each pattern corresponds to a cluster. A strategy to merge couples of clusters is defined as moving up the hierarchy: each cluster in the next level is the fusion of two clusters from the previous level.
2. Divisive, where a single cluster contains all patterns in the first level, then a splitting strategy is defined to halve clusters by moving down the hierarchy.

In this work, the agglomerative hierarchical approach is employed as this is the default MATLAB implementation. The MATLAB algorithm involves three steps:

1. Using a distance metric, find the similarity or dissimilarity between every pair of data points in the dataset;
2. Aggregate data points into a binary hierarchical cluster tree by fusing pairs of clusters according to their distance;
3. Establish the level of the tree where it is cut into k clusters.

The resulting prototypes are selected as the mean vectors of each cluster.

5. Experimental Results

The proposed approach is tested and compared with the canonical approaches with each using the testing protocol proposed by the authors of the datasets. The performance indicator is the classification accuracy and methods were tested on the following animal vocalization datasets:

- BIRDz, which functioned as a control and a real-world audio dataset in [46], a ten-run testing protocol is used; we have used the same split used by the authors of the dataset. The real-world tracks were collected from the Xeno-canto Archive (<http://www.xeno-canto.org/>). BIRDz includes a total of 2762 bird acoustic samples from 11 North American bird species plus 339 “unknown” samples that include noise and unknown species’ vocalizations. The observations are composed of five different spectrograms: 1) constant frequency, 2) frequency modulated whistles, 3) broadband pulses, (4) broadband with varying frequency components, and 5) strong harmonics. The dataset is balanced: the size of all the “bird” classes varies between 246 and 259; only the class “other” is a little larger.
- CAT, [37,47] is a dataset that contains ten balanced classes of approximately 300 samples per class for a total of 2962 samples. The testing protocol is a 10-fold cross-validation. The ten classes represent the following cat vocalizations: (1) Resting, (2) Warning, (3) Angry, (4) Defense, (5) Fighting, (6) Happy, (7) Hunting mind, (8) Mating, (9) Mother call, and (10) Paining. The average duration of each sample is approximately 4 s. Samples were garnered from such online resources as Kaggle, Youtube, and Flickr.

In this section, we report experiments aimed at evaluating the proposed system by varying several components: i.e., the input images (spectrograms or HASC images), the topology of the Siamese Network (NN1, NN2, NN3, NN4), the clustering algorithm (K-Means, K-Medoids, Hierarchical, Spectral), the clustering modality (unsupervised or supervised, i.e., clustering on the whole training set or on each class), and the number of prototypes ($kc = 15, 30, 45, 60$). For the training of the Siamese networks in our experiments, we have selected the training options suggested by the MATLAB framework for Siamese networks. This choice ensures that such values have not been overfitted on the selected dataset. The binary cross-entropy gives the loss function between the predicted score and the true label value. The parameters for ADAM optimization are the following: learning rate 0.0001, gradient decay factor 0.9, squared gradient decay factor 0.99. The number of iterations has been set at 3000 (without any early stop criterion).

In the first experiment, reported in Table 2, only K-means clustering is explored. The results are obtained from the fusion by sum rule of the four SVMs trained using the dissimilarity spaces built with different values for kc (15, 30, 45, 60). Performance is reported for only NN1 and NN2 (the first two network topologies) to reduce the computation time.

The ensembles in Table 2 are obtained by varying the input data (Sp = spectrograms, HASC = HASC images), the type of clustering (unsupervised or supervised), and the network topology. The clustering method is fixed to K-means for all the methods, and the number of prototypes belongs to the following set (15,30,45,60). The column labeled *#classifiers* recaps the number of classifiers in the ensemble, and the first column (“Name”) assigns a name to the ensemble.

The best average performance is obtained by the ensemble FA1_2 in the last row (which is the sum rule of the methods in the first eight rows). On the BIRD dataset, there is a boost in performance with NN1 and NN2 using the HASC images instead of the spectrograms, while on the CAT dataset, HASC images boost the performance of NN2 but not NN1.

The goal of the second experiment is to compare the clustering methods. To achieve this aim, in Table 3, we report the performance of different clustering approaches using the HSup-2 approach (i.e., HASC images as input, NN2 as network topology, and unsupervised version of the clustering). The last row reports the ensemble F_Clu obtained as the sum rule among the above four approaches. F_Clu produces the highest performance, though this gain is only slightly higher than that of K-means. All the clustering algorithms are quite similar in performance. Since their fusion does not achieve a

clear advantage against a single approach, in the next experiments, we use only the K-means strategy for clustering by varying the number of prototypes kc .

Table 2. Performance obtained by k-means clustering: accuracy \pm standard deviation.

Name	Input Image	Network Topology	Clustering Method	Clustering Type	#Prototypes	#Classifiers	CAT	BIRD
Sup-1	Sp	NN1	K-means	S	15, 30, 45, 60	4	78.64 \pm 1.2	92.46 \pm 0.71
Sup-2	Sp	NN2	K-means	S	15, 30, 45, 60	4	76.95 \pm 1.3	92.74 \pm 0.82
UnS-1	Sp	NN1	K-means	U	15, 30, 45, 60	4	81.69 \pm 1.0	92.73 \pm 0.95
UnS-2	Sp	NN2	K-means	U	15, 30, 45, 60	4	75.25 \pm 1.4	92.80 \pm 0.78
HSUp-1	HASC	NN1	K-means	S	15, 30, 45, 60	4	78.64 \pm 1.2	94.52 \pm 0.65
HSUp-2	HASC	NN2	K-means	S	15, 30, 45, 60	4	81.69 \pm 0.9	93.22 \pm 0.82
HUnS-1	HASC	NN1	K-means	U	15, 30, 45, 60	4	79.32 \pm 1.1	94.53 \pm 0.68
HUnS-2	HASC	NN2	K-means	U	15, 30, 45, 60	4	81.36 \pm 1.3	92.97 \pm 0.72
FSp-1	Sp	NN1	K-means	S,U	15, 30, 45, 60	8	81.02 \pm 1.0	92.79 \pm 0.85
FSp-2	Sp	NN2	K-means	S,U	15, 30, 45, 60	8	76.95 \pm 1.2	92.77 \pm 0.76
FA-1	Sp,HASC	NN1	K-means	S,U	15, 30, 45, 60	16	82.37 \pm 0.9	94.50 \pm 0.65
FA2	Sp,HASC	NN2	K-means	S,U	15, 30, 45, 60	16	83.73 \pm 0.9	94.11 \pm 0.70
FA1_2	Sp,HASC	NN1 + NN2	K-means	S,U	15, 30, 45, 60	32	84.41 \pm 0.9	94.37 \pm 0.62

Table 3. Performance obtained considering different clustering algorithms: accuracy \pm standard deviation.

Name	Input Image	Network Topology	Clustering Method	Clustering Type	#Prototypes	#Classifiers	CAT	BIRD
	HASC	NN2	K-means	S	15, 30, 45, 60	4	81.69 \pm 0.9	93.22 \pm 0.82
	HASC	NN2	K-Med	S	15, 30, 45, 60	4	81.02 \pm 1.0	92.85 \pm 0.85
	HASC	NN2	Hier	S	15, 30, 45, 60	4	81.69 \pm 0.9	93.01 \pm 0.87
	HASC	NN2	Spect	S	15, 30, 45, 60	4	80.00 \pm 1.1	93.13 \pm 0.79
F_Clu	HASC	NN2	All	S	15, 30, 45, 60	16	82.03 \pm 0.9	93.37 \pm 0.75

In Table 4, the four network topologies are coupled with K-means clustering and HSUp (i.e., HASC images as input and the unsupervised version of clustering). The last row, which reports the ensemble F_NN is obtained as the sum rule among the above four approaches, and it produces the average best performance on this test.

Table 4. Performance obtained considering different network topologies: accuracy \pm standard deviation.

Name	Input Image	Network Topology	Clustering Method	Clustering Type	#Prototypes	#Classifiers	CAT	BIRD
	HASC	NN1	K-means	S	15, 30, 45, 60	4	78.64 \pm 1.2	94.52 \pm 0.65
	HASC	NN2	K-means	S	15, 30, 45, 60	4	81.69 \pm 1.1	93.22 \pm 0.72
	HASC	NN3	K-means	S	15, 30, 45, 60	4	78.64 \pm 1.2	94.91 \pm 0.64
	HASC	NN4	K-means	S	15, 30, 45, 60	4	82.37 \pm 1.1	93.33 \pm 0.68
F_NN	HASC	All	K-means	S	15, 30, 45, 60	16	84.07 \pm 1.0	94.99 \pm 0.64

Even better results are obtained by combining by sum rule all the approaches reported in the previous tables: the combined performance on CAT is 85.76 and on BIRD 95.08. It is clear that the ensemble strongly outperforms a simple Sup-1. The superiority of one method over another is validated according to the Wilcoxon signed-rank test [54]; F_NN outperforms each of the other methods with a p -value of 0.05.

The following experiment is aimed at evaluating the possibility of creating ensembles by varying some parameters of the method. For a fair comparison among ensembles in Table 5, the performance of

an ensemble obtained by retraining Siamese HSup-1 (i.e., by using the same approach as that reported in Table 2) is compared with ensembles obtained by varying the network topology. The results of Table 5 indicate that changing the network topology introduces diversity in the ensemble: the comparison among ensembles of 4, 8, and 16 networks (see HSup-1x), obtained from the topology NN1, and the ensembles named F_NN, obtained by varying the topology of the Siamese Network, show an evident performance gain in favor of the latter (with the same number of classifiers). It is also interesting to observe the similar results in rows 2 and 3: both ensembles have four networks, but the first is made simply by retraining the same model, while the second has different numbers of prototypes. Thus, varying the values of kc is not very important when building an ensemble. For obtaining superior performance, it is necessary to vary the network topologies.

Table 5. Comparison between ensembles of reiterated Siamese Networks with NN1 and ensembles obtained considering different network topologies: accuracy \pm standard deviation.

Name	Input Image	Network Topology	Clustering Method	Clustering Type	#Prototypes	#Classifiers	CAT	BIRD
HSup-1(1)	HASC	NN1	K-means	S	15	1	75.93 \pm 1.5	93.92 \pm 0.85
HSup-1(4)	HASC	NN1	K-means	S	15	1 \times 4	81.69 \pm 1.3	94.50 \pm 0.78
HSup-1	HASC	NN1	K-means	S	15, 30, 45, 60	4 \times 1	78.64 \pm 1.2	94.52 \pm 0.65
HSup-1(8)	HASC	NN1	K-means	S	15, 30, 45, 60	4 \times 2	80.68 \pm 1.1	94.56 \pm 0.75
HSup-1(16)	HASC	NN1	K-means	S	15, 30, 45, 60	4 \times 4	81.02 \pm 1.0	94.63 \pm 0.77
F_NN(4)	HASC	All	K-means	S	15	4	83.39 \pm 0.9	94.73 \pm 0.62
F_NN(8)	HASC	All	K-means	S	15, 30	8	84.07 \pm 0.8	94.90 \pm 0.60
F_NN	HASC	All	K-means	S	15, 30, 45, 60	16	84.07 \pm 0.8	94.99 \pm 0.58

Finally, reported in Table 6 is a comparison between the Siamese networks and standard CNNs trained with spectrograms. The method labeled eCNN is the fusion among different CNNs (GoogleNet, VGG16, VGG19, and GoogleNetP365).

Table 6. Performance obtained considering different standard CNN.

Method	CAT	BIRD
OLD [43]	82.41	92.97
F_NN	84.07	94.99
GoogleNet	82.98	92.41
VGG16	84.07	95.30
VGG19	83.05	95.19
GoogleNetP365	85.15	92.94
eCNN	87.36	95.81
OLD + eCNN	87.76	95.95
F_NN + eCNN	88.47	96.03

From the results reported in the previous tables, the following conclusions can be drawn:

- The best way for building an ensemble of Siamese networks is to combine different network topologies;
- The proposed F_NN ensemble improves previous methods based on Siamese networks (cf. OLD in Table 6);
- F_NN obtains a performance that is similar to eCNN on BIRD but lower than eCNN on CAT;
- The best performance in both datasets is gained by sum rule between eCNN and F_NN (i.e., the fusion among CNNs and the Siamese networks).

As far as the computation requirements are concerned, using a Xeon E5-1603 v4-2.8 GHz-64 GB Ram, the proposed approach requires a classification time of less than 0.028 s per a batch of 1251

images in the testing phase, the creation of the test dissimilarity space requires 44.30 s per a batch of 1251 images, and Hasc needs 0.006 s to be performed in a given spectrogram. The training phase for a training dataset of 1511 images requires 4415 s for a single NN1 training (using a single GPU Titan Xp), 88.60 s for the creation of a single training dissimilarity space, and 0.07 s for the training of a single SVM. Thus, the time required for testing is near real-time using a powerful server/GPU.

To further validate our approach, we tested it on the ESC-50 benchmark audio classification dataset. To reduce the computation time, only Sup-1 was tested. It obtained 52% accuracy but needed a high number of training iterations for network convergence. For the ESC-50 dataset, Sup-1 was trained for 25,000 epochs rather than for 3000 epochs, as was done for CAT and BIRD. For comparison, a simple CNN with only 3-layers [55] obtained an accuracy of 54%.

In Table 7, some other state-of-the-art results in the literature are reported on the CAT and BIRD datasets (using the same testing protocol). As can be observed, the performance of the ensembles described in this paper approaches those reported in the literature. Note that in Table 7 the results from [46] are based on a feature selection approach where the number of selected features is the hyperparameters selected on that dataset.

Table 7. Literature results.

Authors	Reference	CAT	BIRD
Nanni et al.	[56]	—	96.3
Nanni et al.	[2]	—	95.1
Zhao et al.	[7]	—	93.6
Pandeya & Lee.	[47]	87.7	—
Pandeya et al.	[37]	91.1	—
Pandeya et al.	[37]–CNN	90.8	—
Zhang et al.	[46]	—	96.7

Finally, it is worth noting that for a more fair comparison among acoustic animal classification works, a deeper experimental evaluation across multiple datasets is needed. The experiments presented in this paper speak to the robustness of the proposed approach: competitive classification accuracy, compared to the state-of-the-art in the literature, has been obtained on two datasets including very different data without any ad-hoc parameter tuning. These results were produced by following a clear and unambiguous testing protocol. The value of reporting methods across datasets means that the results reported here can reasonably serve as a baseline for later research comparisons in this area.

6. Conclusions

This work presents a method for classifying animal vocalizations using four Siamese networks and dissimilarity spaces. Different clustering techniques taking both a supervised and unsupervised approach were used for dissimilarity space generation. A compact descriptor is obtained by projecting each pattern into the dissimilarity spaces generated by the clustering methods using different numbers of centroids and the outputs of the four Siamese networks. The classification step is performed using a set of SVMs trained from such descriptors and combined by the sum rule to obtain a highly competitive ensemble as tested on two datasets of animal vocalizations. In addition, experimental results demonstrated the diversity between the proposed approach and other state-of-the-art approaches can be exploited in an ensemble to further improve the classification performance. The fusions improved the performance on both audio classification problems, outperforming the standalone approaches.

Future work in this area will focus on experimentally deriving ensembles using the same approach. The goal will be to assess the approach proposed here for generalizability across many sound classification problems, such as those cited in [36,44]. This will involve testing the proposed method by adding more supervised and unsupervised clustering techniques and additional Siamese network architectures.

Author Contributions: L.N. conceived of the presented idea, G.M., L.N., A.L. performed the experiments. S.B., A.L. wrote the manuscript. S.B. provided some resources. All authors have read and agree to the published version of the manuscript.

Funding: This research received no external funding.

Acknowledgments: The authors are grateful to NVIDIA Corporation for supporting this research with the donation of a Titan Xp GPU.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Padmanabhan, J.; Premkumar, M.J.J. Machine learning in automatic speech recognition: A survey. *IETE Tech. Rev.* **2015**, *32*, 240–251. [\[CrossRef\]](#)
2. Nanni, L.; Costa, Y.M.; Lucio, D.R.; Silla, C.N., Jr.; Brahnam, S. Combining visual and acoustic features for audio classification tasks. *Pattern Recognit. Lett.* **2017**, *88*, 49–56. [\[CrossRef\]](#)
3. Sahoo, S.; Choubisa, T.; Prasanna, S.M. Multimodal Biometric Person Authentication: A Review. *IETE Tech. Rev.* **2012**, *29*, 54–75. [\[CrossRef\]](#)
4. Li, S.; Li, F.; Tang, S.; Xiong, W. A Review of Computer-Aided Heart Sound Detection Techniques. *BioMed Res. Int.* **2020**, *2020*, 5846191. [\[CrossRef\]](#)
5. Chandrakala, S.; Jayalakshmi, S.L. Generative Model Driven Representation Learning in a Hybrid Framework for Environmental Audio Scene and Sound Event Recognition. *IEEE Trans. Multimed.* **2019**, *22*, 3–14. [\[CrossRef\]](#)
6. Chachada, S.; Kuo, C.-C.J. Environmental sound recognition: A survey. In Proceedings of the 2013 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference, Kaohsiung, Taiwan, 29 October–1 November 2013; pp. 1–9. [\[CrossRef\]](#)
7. Zhao, Z.; Zhang, S.H.; Xu, Z.Y.; Bellisario, K.; Dai, N.H.; Omrani, H.; Pijanowski, B.C. Automated bird acoustic event detection and robust species classification. *Ecol. Inform.* **2017**, *39*, 99–108. [\[CrossRef\]](#)
8. Badshah, A.M.; Ahmad, J.; Rahim, N.; Baik, S.W. Speech emotion recognition from spectrograms with deep convolutional neural network. In Proceedings of the 2017 International Conference on Platform Technology and Service (PlatCon), Busan, Korea, 13–15 February 2017; pp. 1–5.
9. Zeng, Y.; Mao, H.; Peng, D.; Yi, Z. Spectrogram based multi-task audio classification. *Multimed. Tools Appl.* **2019**, *78*, 3705–3722. [\[CrossRef\]](#)
10. Lidy, T.; Rauber, A. Evaluation of feature extractors and psycho-acoustic transformations for music genre classification. In Proceedings of the 6th International Conference on Music Information Retrieval, London, UK, 11–15 September 2005; pp. 34–41.
11. Wyse, L. Audio spectrogram representations for processing with convolutional neural networks. *arXiv* **2017**, arXiv:1706.09559.
12. Rubin, J.; Abreu, R.; Ganguli, A.; Nelaturi, S.; Matei, I.; Sricharan, K. Classifying heart sound recordings using deep convolutional neural networks and mel-frequency cepstral coefficient. In Proceedings of the Computing in Cardiology (CinC), Vancouver, BC, Canada, 11–14 September 2016.
13. Nanni, L.; Costa, Y.M.G.; Brahnam, S. Set of texture descriptors for music genre classification. In Proceedings of the 22nd WSCG International Conference on Computer Graphics, Visualization and Computer Vision, Plzen, Czech Republic, 2–5 June 2014.
14. Haralick, R.M. Statistical and structural approaches to texture. *Proc. IEEE* **1979**, *67*, 786–804. [\[CrossRef\]](#)
15. Ojansivu, V.; Heikkila, J. Blur insensitive texture classification using local phase quantization. In Proceedings of the ICISP, Cherbourg-Octeville, France, 1–3 July 2008.
16. Ojala, T.; Pietikainen, M.; Maenpaa, T. Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. *IEEE Trans. Pattern Anal. Mach. Intell.* **2002**, *24*, 971–987. [\[CrossRef\]](#)
17. Brahnam, S.; Jain, L.C.; Lumini, A.; Nanni, L. (Eds.) *Local Binary Patterns: New Variants and Applications*; Springer: Berlin, Germany, 2014.
18. Costa, Y.M.G.; Oliveira, L.S.; Koerich, A.L.; Gouyon, F.; Martins, J.G. Music genre classification using LBP textural features. *Signal Process.* **2012**, *92*, 2723–2737. [\[CrossRef\]](#)
19. Costa, Y.M.G.; Oliveira, L.S.; Koerich, A.L.; Gouyon, F. Music genre recognition using spectrograms. In Proceedings of the 18th International Conference on Systems, Signals and Image Processing, Sarajevo, Bosnia and Herzegovina, 16–18 June 2011.

20. Costa, Y.M.G.; Oliveira, L.S.; Koerich, A.L.; Gouyon, F. Music genre recognition using gabor filters and LPQ texture descriptors. In Proceedings of the 18th Iberoamerican Congress on Pattern Recognition, Havana, Cuba, 20–23 November 2013.
21. Ren, Y.; Cheng, X. Review of convolutional neural network optimization and training in image processing. In Proceedings of the 10th International Symposium on Precision Engineering Measurements and Instrumentation (ISPEMI 2018), Kunming, China, 8–10 August 2018; SPIE: Bellingham, WA, USA, 2019.
22. Wang, X.; Zhao, Y.; Pourpanah, F. Recent advances in deep learning. *Int. J. Mach. Learn. Cybern.* **2020**, *11*, 747–750. [\[CrossRef\]](#)
23. Humphrey, E.; Bello, J.P. Rethinking automatic chord recognition with convolutional neural networks. In Proceedings of the International Conference on Machine Learning and Applications, Boca Raton, FL, USA, 12–15 December 2012.
24. Humphrey, E.; Bello, J.P.; LeCun, Y. Moving beyond feature design: Deep architectures and automatic feature learning in music informatics. In Proceedings of the International Conference on Music Information Retrieval, Porto, Portugal, 8–12 October 2012; pp. 403–408.
25. Nakashika, T.; Garcia, C.; Takiguchi, T. Local-feature-map integration using convolutional neural networks for music genre classification. In Proceedings of the Interspeech 2012 13th Annual Conference of the International Speech Communication Association, Portland, OR, USA, 9–13 September 2012; pp. 1752–1755.
26. Costa, Y.M.; Oliveira, L.S.; Silla, C.N., Jr. An evaluation of Convolutional Neural Networks for music classification using spectrograms. *Appl. Soft Comput.* **2017**, *52*, 28–38. [\[CrossRef\]](#)
27. Sigtia, S.; Dixon, S. Improved music feature learning with deep neural networks. In Proceedings of the IEEE International Conference on Acoustic, Speech and Signal Processing, Florence, Italy, 4–9 May 2014.
28. Wang, C.Y.; Santoso, A.; Mathulapransan, S.; Chiang, C.C.; Wu, C.H.; Wang, J.C. Recognition and retrieval of sound events using sparse coding convolutional neural network. In Proceedings of the IEEE International Conference on Multimedia and Expo (ICME), Hong Kong, China, 10–14 July 2017.
29. Oramas, S.; Nieto, O.; Barbieri, F.; Serra, X. Multilabel music genre classification from audio, text and images using deep features. In Proceedings of the International Society for Music Information Retrieval (ISMIR) Conference, Suzhou, China, 23–27 October 2017.
30. Kong, Q.; Xu, Y.; Sobieraj, I.; Wang, W.; Plumbley, M.D. Sound Event Detection and Tim Frequency Segmentation from Weakly Labelled Data. *IEEE ACM Trans. Audio Speech Lang. Process.* **2019**, *27*, 777–787. [\[CrossRef\]](#)
31. Nanni, L.; Brahnam, S.; Lumini, A.; Barrier, T. Ensemble of local phase quantization variants with ternary encoding. In *Local Binary Patterns: New Variants and Applications*; Brahnam, S., Jain, L.C., Lumini, A., Nanni, L., Eds.; Springer: Berlin, Germany, 2014; pp. 177–188.
32. Cao, Z.; Principe, J.C.; Ouyang, B.; Dalgleish, F.; Vuorenkoski, A. Marine animal classification using combined CNN and hand-designed image features. In Proceedings of the MTS/IEEE Oceans, Washington, DC, USA, 19–22 October 2015.
33. Salamon, J.; Bello, J.P.; Farnsworth, A.; Kelling, S. Fusing shallow and deep learning for bioacoustic bird species. In Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), New Orleans, LA, USA, 5–9 March 2017.
34. Cullinan, V.I.; Matzner, S.; Duberstein, C.A. Classification of birds and bats using flight tracks. *Ecol. Inform.* **2015**, *27*, 55–63. [\[CrossRef\]](#)
35. Acevedo, M.A.; Corrada-Bravo, C.J.; Corrada-Bravo, H.; Villanueva-Rivera, L.J.; Aide, T.M. Automated classification of bird and amphibian calls using machine learning: A comparison of methods. *Ecol. Inform.* **2009**, *4*, 206–214. [\[CrossRef\]](#)
36. Fristrup, K.M.; Watkins, W.A. *Marine Animal Sound Classification*; WHOI Technical Reports; Woods Hole Oceanographic Institution: Woods Hole, MA, USA, 1993; Available online: <https://hdl.handle.net/1912/546> (accessed on 30 October 2020).
37. Pandeya, Y.R.; Kim, D.; Lee, J. Domestic cat sound classification using learned features from deep neural nets. *Appl. Sci.* **2018**, *8*, 1949. [\[CrossRef\]](#)
38. Wang, A. An industrial strength audio search algorithm. In Proceedings of the ISMIR Proceedings, Baltimore, MD, USA, 26–30 October 2003.
39. Haitisma, J.; Kalker, T. A Highly Robust Audio Fingerprinting System. In Proceedings of the ISMIR, Paris, France, 13–17 October 2002.

40. Manocha, P.; Badlani, R.; Kumar, A.; Shah, A.; Elizalde, B.; Raj, B. Content-based representations of audio using siamese neural networks. In Proceedings of the 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Calgary, AB, Canada, 15–20 April 2018; pp. 3136–3140.
41. Droghini, D.; Vesperini, F.; Principi, E.; Squartini, S.; Piazza, F. Few-shot siamese neural networks employing audio features for human-fall detection. In Proceedings of the International Conference on Pattern Recognition and Artificial Intelligence, Union, NJ, USA, 15–17 August 2018. [\[CrossRef\]](#)
42. Zhang, Y.; Pardo, B.; Duan, Z. Siamese Style Convolutional Neural Networks for Sound Search by Vocal Imitation. *IEEE/ACM Trans. Audio, Speech, Lang. Process.* **2018**, *27*, 429–441. [\[CrossRef\]](#)
43. Nannia, L.; Rigo, A.; Lumini, A.; Brahnam, S. Spectrogram Classification Using Dissimilarity Space. *Appl. Sci.* **2020**, *10*, 4176. [\[CrossRef\]](#)
44. Agrawal, A. Dissimilarity learning via Siamese network predicts brain imaging data. *arXiv* **2019**, arXiv:1907.02591.
45. Bromley, J.; Bentz, J.W.; Bottou, L.; Guyon, I.; LeCun, Y.; Moore, C.; Säckinger, E.; Shah, R. Signature verification using a Siamese time delay neural network. *Int. J. Pattern Recognit. Artif. Intell.* **1993**, *7*, 669–688. [\[CrossRef\]](#)
46. Zhang, S.H.; Zhao, Z.; Xu, Z.Y.; Bellisario, K.; Pijanowski, B.C. Automatic bird vocalization identification based on fusion of spectral pattern and texture features. In Proceedings of the 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Calgary, AB, Canada, 15–20 April 2018; pp. 271–275.
47. Pandeya, Y.R.; Lee, J. Domestic Cat Sound Classification Using Transfer Learning. *Int. J. Fuzzy Log. Intell. Syst.* **2018**, *18*, 154–160. [\[CrossRef\]](#)
48. Biagio, M.S.; Crocco, M.; Cristani, M.; Martelli, S.; Murino, V. Heterogeneous auto-similarities of characteristics (hasc): Exploiting relational information for classification. In Proceedings of the IEEE Computer Vision (ICCV13), Sydney, Australia, 3–6 December 2013.
49. Piczak, K.J. ESC: Dataset for Environmental Sound Classification. In Proceedings of the 23rd ACM international conference on Multimedia, Brisbane, Australia, 26–30 October 2015. [\[CrossRef\]](#)
50. Vapnik, V. The support vector method. In Proceedings of the Artificial Neural Networks ICANN'97, Lausanne, Switzerland, 8–10 October 1997.
51. Chicco, D. Siamese neural networks: An overview. In *Artificial Neural Networks. Methods in Molecular Biology*; Cartwright, H., Ed.; Springer Protocols; Humana: New York, NY, USA, 2020; Volume 2190, pp. 73–94.
52. Glorot, X.; Bordes, A.; Bengio, Y. Deep Sparse Rectifier Neural Networks. In Proceedings of the AISTATS, Ft. Lauderdale, FL, USA, 11–13 April 2011; Available online: https://pdfs.semanticscholar.org/6710/7f78a84bdb2411053cb54e94fa226eea6d8e.pdf?_ga=2.211730323.729472771.1575613836-1202913834.1575613836 (accessed on 30 October 2020).
53. Maas, A.L. Rectifier Nonlinearities Improve Neural Network Acoustic Models. 2013. Available online: https://pdfs.semanticscholar.org/367f/2c63a6f6a10b3b64b8729d601e69337ee3cc.pdf?_ga=2.208124820.729472771.1575613836-1202913834.1575613836 (accessed on 30 October 2020).
54. Demšar, J. Statistical comparisons of classifiers over multiple data sets. *J. Mach. Learn. Res.* **2006**, *7*, 1–30.
55. Huzaifah, M. Comparison of Time-Frequency Representations for Environmental Sound Classification using Convolutional Neural Networks. *arXiv* **2017**, arXiv:1706.07156.
56. Nanni, L.; Costa, Y.; Lumini, A.; Kim, M.Y.; Baek, S.R. Combining visual and acoustic features for music genre classification. *Expert Syst. Appl.* **2016**, *45*, 108–117. [\[CrossRef\]](#)

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.



© 2020 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<http://creativecommons.org/licenses/by/4.0/>).