EVALUATION OF DEEP LEARNING TECHNIQUES FOR ACOUSTIC ENVIRONMENTAL EVENTS DETECTION

SVETLANA SEGĂRCEANU[1], INGE GAVĂŢ[2,], GEORGE SUCIU[1]

Abstract. Environmental sounds detection plays an essential role in computer science and robotics as it simulates the human function of hearing. It is applied in environment research, monitoring and protection, by allowing investigation of natural reserves, and showing potential risks of damage deduced from the environmental acoustic. In this paper we present experimental results of attempts to separate different types of acoustic events from continuous environment recordings. We apply some deep learning approaches i.e., Deep Feed Forward networks, and Long short-term memory (LSTM) recurrent neural networks, feeding at input several types of parameters: spectral, cepstral and temporal features, and assessing various input/output data organization, or network configuration. The methods are evaluated and compared with some classical methods explored before.

Key words: environmental sound recognition, deep neural networks, spectral features.

1. INTRODUCTION

Environmental noise recognition became an essential field of computer science and robotics, as it simulates the important function of human hearing, and is intended to overpower it. Hearing, the second most important human sense after vision, is not to be ignored when building a computer. Until recently, the research in acoustic signal processing has been done mostly around speech and music, with less exploration for the non-speech environmental sounds. However, lately, other categories of sounds receive attention as well, in the framework of a new discipline AESR (Automatic Environment Sound Recognition) with applications in content-based search, robotics [1], security environment monitoring and protection [2–5]. It involves the study of such aspects as AED (Acoustic Event Detection) or computational auditory scene analysis (CASA). By environmental sounds, we mean everyday sounds, natural (leaves rustling, animal noise, birds chirping, water

[1] Beia Consult International, Bucharest, Romania
[2] Politehnica University of Bucharest, Department of applied Electronics and information engineering,

ripple, wind blowing through trees, waves crashing onto the shore, thunder) or artificial (produced by engines like cars, cranes, ATVs, snowmobile, lawnmower, pneumatic hammer, chainsaw, or creaky door, gunshots, crowd in club).

In the framework of the SeaForest project, we proposed an architecture based on a sensor network. The design of the framework includes:

- A wireless network of sensors to collect data from the forest environment, for monitoring gas emissions, heat, humidity acoustic data;
- A data processing module based on a MQTT server application.
- An acoustic data processing module to identify nature of acquired signals.

The focus of our research in the context of forest monitoring is logging risk detection by identification of specific classes of sounds: chainsaw, vehicles, or maybe speech. We extend an earlier research on acoustic signal processing, by exploring the state-of-the-art paradigm in data modelling and, the Deep Neural Networks (DNN). We examine two types of DNNs, the Deep Feed Forward Neural Networks (FFNN) and a popular version of Recurrent Neural Networks (RNNs), the Long Short-Term Memory (LSTM). The two neural networks will be run on two types of feature spaces: the Mel-cepstral and the Fourier power spectrum feature spaces. We will compare the achieved results with the former performance obtained using the Gaussian Mixtures Modelling (GMM) and the Dynamic Time Warping (DTW). Another purpose of the paper is to clarify some issues concerning signal pre-processing framework, like length of the analysis window and the underlying frequency domain to be used in spectral analysis.

The paper is structured as follows: next the state-of-the-art in environmental sound recognition is described; the third section details our approach: signal feature extraction and modelling methods we applied in sound recognition; the fourth section presents the setup of the experiments and evaluates the proposed methods; the last part lays the conclusions of the paper.

2. STATE-OF-THE-ART

Early attempts to assess speech typical methods in the context of non-speech acoustics can be found for instance in [6, 7]. Artificial Neural Networks (ANN), DTW, Hidden Markov Models (HMM), Learning Vector Quantization (LVQ), applied on Fourier or Linear Predictive Coding (LPC) features are evaluated.

In [8] an overall investigation of different categories of acoustic signals and the appropriate recognition methodologies. The environmental sounds are classified as stationary and non-stationary. The framework for stationary acoustic signals coincides to a great extent with the one used in voice-based applications (speech or speaker recognition) in what concerns features extraction, where spectral features prevail, derived from Mel analysis, LPC, Spectral Dynamic Features (SDF), techniques based on signal autocorrelation, and feature space modelling methods, where GMM, k-Nearest Neighbours (k-NN), LVQ, DTW,

HMM, Support Vector Machine (SVM), DNNs are frequently applied. Concerning the non-stationary signals, successful methods are based on sparse representations using the Matching Pursuit (MP) to generate MP-Gabor features. Alternative approaches use fusion of different methods to boost the performance of the system.

In [9] the authors evaluate the up-to-the-minute methodologies used in AESR. The leading approaches are GMM, in fact, the GMM-UBM (Universal Background Model). SVM and two neural network architectures, DNN/RNN, in the context of open-set identification experiments on two types of acoustic events, baby cries and smoke alarm, and a very large world set of ambient sounds as source of impostor, representing a great range of complementary acoustic events. The FFNN yields the best identification rate, while the best computational cost is achieved by GMM. SVM has an intermediate performance as identification rate yet at a very high computational cost.

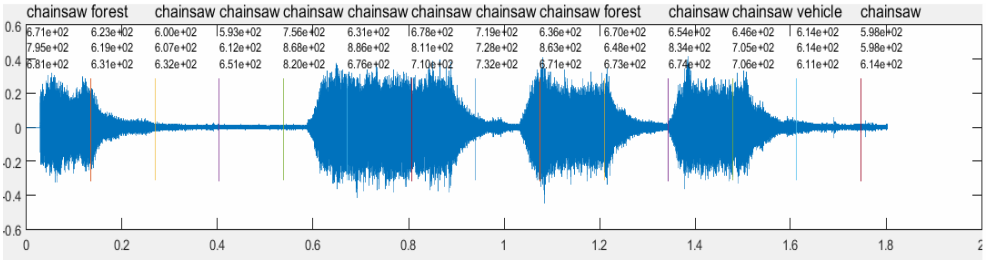

Fig. 1 – The framing process of a real-life sound with classification of each 3s frame.

Concerning the framework applied in AESR, this material draws on the ideas presented in [8]. The usual pre-processing applied in AESR includes a framing process for the acoustic signal, usually followed by sub-framing or sequential processing. In the “framing” stage the signal is processed continuously, frame by frame. A classification decision is made for each frame and successive frames may belong to different classes. This can enhance the acoustic signal classification by structuring the stream into more homogeneous blocks to better catch the acoustic event. In real circumstances it could answer the questions "What happened when?" and “How persisting and possibly dangerous the happening events are?”. Yet, there is no way of setting an optimal frame length, as for some acoustic events, such as thunder or gunshot, the window length suited for engine detection, might be too large, and contain several other acoustic events. Thus, certain events could be associated to inappropriate classes. For applications like chainsaw detection in forest environment we consider a length of about 2–3 s. This is illustrated in Fig. 1, where a chainsaw is detected in a forest environment. The scores are obtained by applying the GMM approach, for three classes of sounds: chainsaw, vehicle, genuine forest.

The latest advances in instrumentation allow applying different frame lengths during monitoring, based on detecting signal energy layers of the environmental

sounds, to spare energy consumption. For low energy layers the length of the frame is increased and it is decreased when a rise of the energy layer is detected.

Next, usually a sub-framing approach is applied, by dividing the frame into, usually overlapping, analysis subframes. The length of a subframe is explicitly set in [8] to 20–30 ms. This length is suited for the analysis of speech sounds as it ensures a good resolution in time and frequency. Fig. 2 presents 22 ms of a male speech which includes three fundamental periods of the respective voice. Whereas Fig. 3 represents 22 ms of chainsaw sound, Fig. 4, 44 ms of chainsaw, and Fig. 5, 88 ns of the same sound. From the image in the Fig. 3 we cannot infer anything about the periodicity of the chainsaw sound, while the two other intervals contain two respectively four periods of the sound. Therefore, considering sub-frames of over 44 ms is a reasonable solution for detecting chainsaw activities. On the other hand, in an acoustical environment different frame lengths could be suited to different acoustic events, so setting the analysis frame length should be a compromise between different lengths suited to different events.

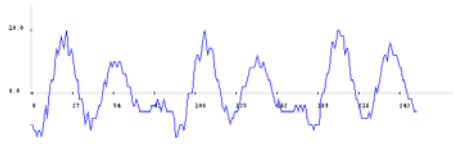

Fig. 2 –. 22 ms of male speech.

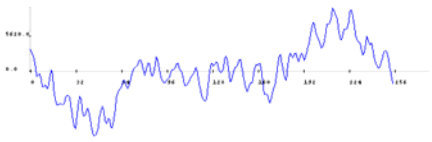

Fig. 3 – 22 ms of a chainsaw sound.

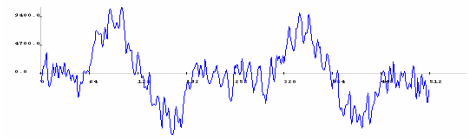

Fig. 4 – 44 ms of a chainsaw sound.

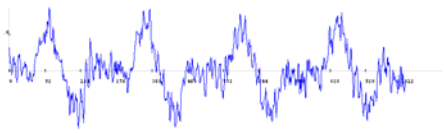

Fig. 5 – 88 ms of a chainsaw sound.

The general processing applied on the analysis frames is usually the same with the one applied in speech signal applications and its final goal is to extract characteristic features. The largely applied features are related to the spectral analysis, i.e., derived from the Fourier features, like the Fourier coefficients, or cepstral features (MFCCs). Non spectral features are energy, Zero-Crossing Rate (ZCR), Spectral Flatness (SF), calculated in temporal domain. Concerning spectral analysis, the relevant frequency interval, for a signal sampling frequency of 44.1 kHz is not the whole frequency domain [0, 22.05] kHz, but a shorter range, as shown in Figs. 5, 6. By setting the appropriate value for the frequency interval, we may improve the performance as accuracy and speed of execution, as another benefit of shortening the frequency interval is the decrease of the number of spectrum samples to be processed.

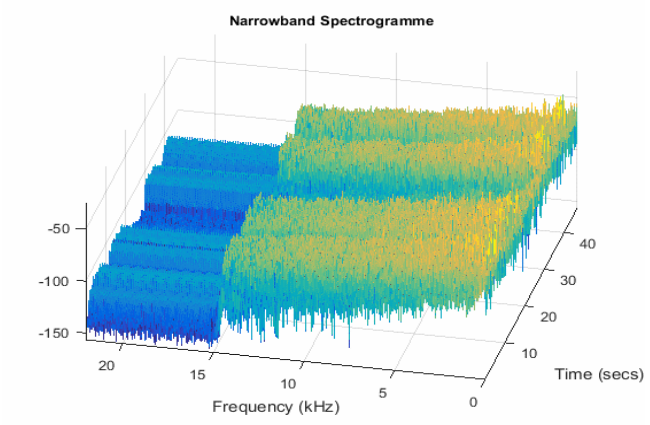

Fig. 6 – Power spectrum of a chainsaw sound.

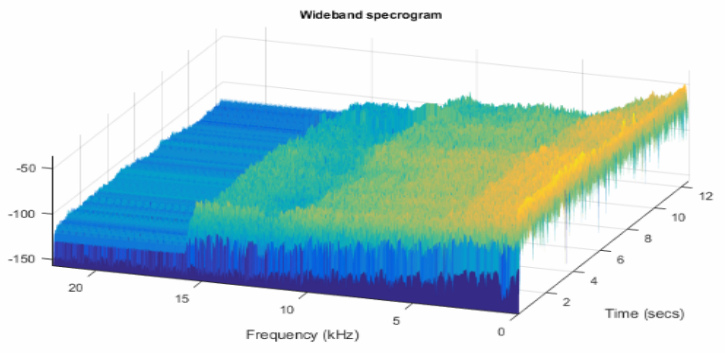

Fig. 7 – Power spectrum of a snowmobile sound.

The usual pre-processing is applied on the analysis frame, and it includes (hamming) windowing and pre-emphasis.

Classification methodology employed in environmental sound recognition is in many concerns similar to the one applied in speaker recognition. It might be aimed to either identification of environmental sounds, like in speaker identification, or to confirm or deny the nature of a sound like in speaker verification operated by authentication programs.

3. THE METHOD

Our methodology is inspired by the above presented facts. As features we used the Mel-cepstral coefficients [10], accompanied or not by SF and ZCR [11],

and the Fourier power spectrum. Modeling the feature space was accomplished using the DNN architectures: FFNN and LSTM.

3.1. FEED FORWARD DEEP NEURAL NETWORKS

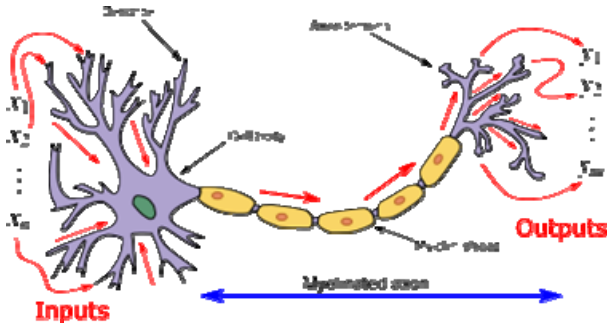

Fig. 8– Neuron and myelinated axon, with signal flow from inputs at dendrites to outputs at axon terminals.

The artificial neural networks (ANNs) were designed to simulate human associative memory, as presented in Fig. 8 [12]. They learn by processing known input examples, and corresponding expected results, creating weighted associations between them, stored within the network data structure. The input set consists of multiple independent variables, rather than a single value. Learning from a given example is made by difference in the state of the net before and after processing the example, leading to estimation of weighted relationships.

3.1.1. Architecture of artificial neural networks

ANNs are composed of artificial neurons, which express the biological concept of neurons. They receive input data, combine the input through internal processing elements like weights and bias terms, and apply an optional threshold using an activation (transfer) function, as shown in Fig. 8.

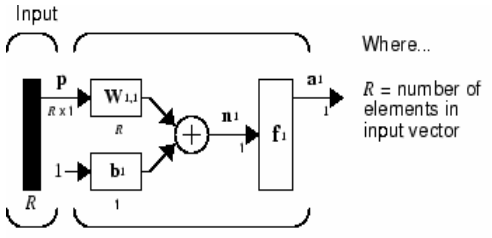

Fig. 9 – Structure of a neuron.

Transfer functions provide a smooth, differentiable transition as input values change, i.e., small changes in input produce small changes in output. They are used to model the output to lie between yes and no, mapping the output values between 0 to 1 or –1 to 1, etc. Transfer functions are basically divided into:

- Linear Activation Functions,
- Non-linear Activation Functions, usually S – shaped functions such as: Logistic (a.k.a. Sigmoid or Soft step):

$$f(x)=\sigma(x)=\frac{1}{1+e^{-x}}, \tag{1}$$

hyperbolic tangent, *arctg* function or Rectified linear unit (ReLU)

$$f(x)=\sigma(x)=\max(0,x). \tag{2}$$

Input data to artificial neurons are usually multidimensional pre-processed numerical data coming from text documents, images, acoustic environment.

Deep feedforward networks (FFNN), also called feedforward neural networks [13], [14], or multilayer perceptrons (MLPs), are the quintessential deep learning models. The goal of a feedforward network for modelling and classification is to define a mapping $y=f(x,\theta)$ and learn the value of parameters θ to ensure the best approximation of the expected value y by the output of f, given the input x and parameters θ. FF have one or more hidden layers of sigmoid neurons followed by an output layer of linear neurons. The general diagram is shown in Figs. 10, 11, where the parameters to be tuned are the weight matrices and bias terms applied at the input of each layer, so that at the output of the overall system would be close to expected values. These networks are called feedforward because the information flows in one direction through intermediate computations and there is no feedback connection. The number of neurons does not necessarily decrease with the layer layer as presented in Fig. 10, but usually the goal is to reduce the dimensionality of the input layer, a process similar to feature extraction. A layer of neurons can be expressed by matrix of weights and bias vectors, as in Fig. 11. The transfer function is supposed to be the same for each neuron.

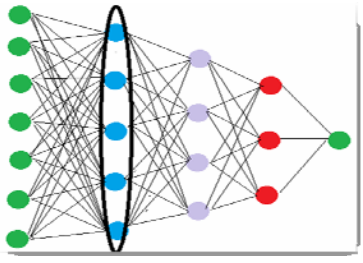

Fig. 10 – Feed Forward Neural Network architecture.

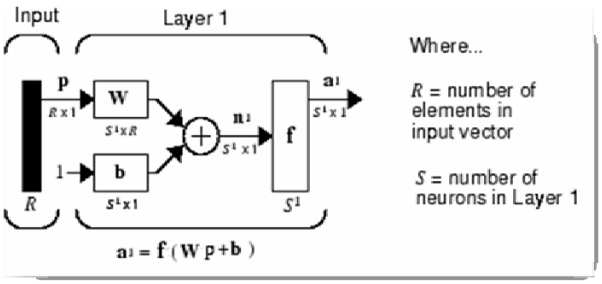

Fig. 11– Structure of a layer of neurons.

The computation corresponding to Figs. 10, 12 can be written as

$$a^4 == f^4\left(W_4\left(f^3\left(W_3\left(f^2\left(W_2\left(f^1\left(W_1 p+b_1\right)\right)+b_2\right)\right)+b_3\right)\right)+b_4\right), \tag{3}$$

or more generally:

$$\begin{aligned} a^k &= f^k\left(W_k a^{k-1}+b_k\right)=f^k\left(W_k\left(f^{k-1}\left(W_{k-1}a^{k-2}+b_{k-1}\right)\right)+b_k\right)= \\ &= f^k\left(W_k\left(f^{k-1}\left(W_{k-1}\left(f^{k-2}\left(W_{k-2}a^{k-3}+b_{k-2}\right)\right)+b_{k-1}\right)\right)+b_k\right)=\ldots . \end{aligned} \tag{4}$$

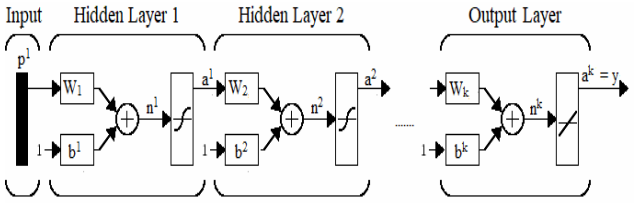

Fig. 12 – Flow of data in a feedforward network.

In the equations above the known information is

a) The input parameters p. The input might be:

- Measurements from sensors: wind speed, temperature, humidity
- Parameters coming from images: matrices of colours, or grey hues.
- Parameters coming from acoustic signals: Fourier spectrum on an analysis

window, or more complicated parameters like cepstral, lineal prediction.

b) The expected output: for instance, if we want to solve a three classes problem the corresponding output for each class input would be either:

- Unidimensional (a scalar value for each class): $(-1,0,1)$ or $(0,1,2)$.

- Multidimensional (a vector for each class): $(1,0,0)$, $(0,1,0)$, $(0,0,1)$.

c) The neural network architecture: number of hidden layers, number of neurons on each layer, etc.

Unknown parameters are:

a) weights: W_k,

b) bias terms: b_k.

3.1.2. Training

Training of a Neural Network can be made in two ways: in batch mode and in incremental mode [13]. In batch mode weights and biases are only updated after all the inputs and targets are presented. The method can be applied to both static and dynamic networks. Incremental Networks receive the inputs one by one and adapt the weights according to each input. Usually, batch training is used. Equation (5) has several unknowns, the weight matrices and the bias terms, and a much larger training data, all the input data and the corresponding expected values. This implicates the realistic conclusion that there will not be any solution of the equation system. All we can do would be to find those parameters, weights and biases, that make the error between the output value and expected output, minimal:

$$e(W,b)=\sum_{i=1}^{N}\left(y_i-a_i^k\left(p,W,b\right)\right)^2, \tag{5}$$

where N is the number of (input, output) pair samples.

To minimize the least mean square expression the LMS algorithm or Widrow-Hoff learning scheme, based on an approximate steepest descent procedure, are used. Adaptive linear networks are trained on examples of correct behaviour. There are many algorithms used to solve the equation. Mathlab has implemented and supports many network training algorithms: Levenberg-Marquardt Algorithm (LMA), Bayesian Regularization (BR), BFGS Quasi-Newton, Resilient Backpropagation, Scaled Conjugate Gradient, One Step Secant, etc. To start minimization using any of these the user should provide an initial guess for the parameter vector $\theta=(W,b)$. The performance of the system depends on this initial guess. Most of the above algorithms try to optimize this process.

3.1.3. Classification

At the end of the training process, we get a FFNN model, (W_k,b_k), $k=1,2,...K$, where K is the number of layers in the network. To classify a

sequence of data we “feed” them at the input of the network, perform all the operations applying the weights and biases to the input data, as in Fig. 12, and evaluate the output. If we coded the output classes as $\{y_1, y_2, ..., y_C\}$, C the number of classes, we compare the obtained output to these values and if the output is closest to y_c the input data will belong to class c.

3.2. LONG SHORT-TIME MEMORY

LSTM is an artificial recurrent neural network architecture RNNs are designed to handle sequences of events that occur in succession, with the understanding of each event based on information from previous events and are able to handle tasks such as such stock prediction or enhanced speech detection. One significant challenge for RNNs performance is that of the vanishing gradient which impacts RNNs long-term memory capabilities, thus are restricted to only remembering a few sequences at a time. LSTMs proposes an architecture to overcome the vanishing gradient problem and allow retain information for longer periods compared to traditional RNNs. Unlike standard feedforward neural networks, LSTM has feedback connections. It is capable of learning long-term dependencies, useful for certain types of prediction requiring the network to retain information over longer time periods, can process entire sequences of data (such as speech or video).

It was introduced in 1997 by the German researchers, Hochreiter and Schmidhuber.

3.2.1. Architecture

Common architecture includes a cell (the memory part of the LSTM unit) and three "regulators", called gates, of the flow of information inside the LSTM unit:

- input gate to control the extent to which new values flow into the cell,
- output gate to control the extent to which a value remains in the cell,
- forget gate to control the extent to which the value in the cell is used to compute the output activation of the LSTM unit.

The LSTM is able to remove or add information to the cell state, through these gates. Some variations of the LSTM unit ignore one or more of these gates.

- Peephole LSTM
- Convolutional LSTM

The cell is responsible for keeping track of the dependencies between the elements in the input sequence. The activation function of the LSTM gates is often the logistic sigmoid function. The connections to and from the LSTM gates, some

recurrent, are weighted. The weights are learned during training, they determine how the gates operate. The diagram of a cell is presented in Fig. 13 [15] and the LSTM flow is shown in Fig. 14.

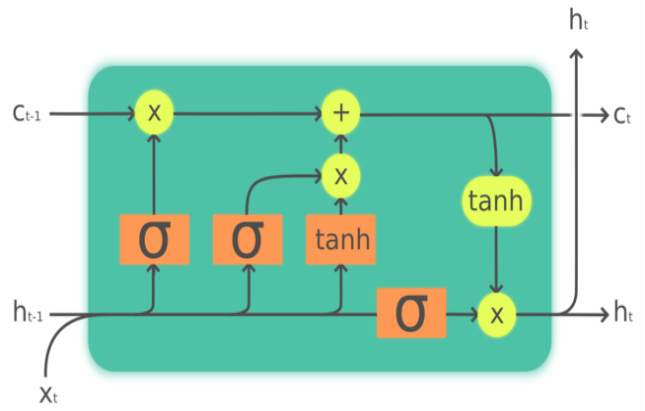

Fig. 13– Structure of a LSTM cell.

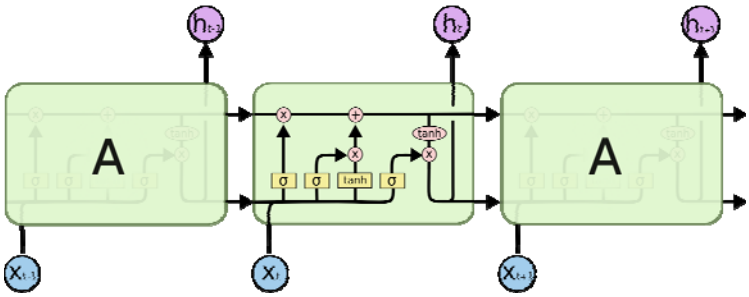

Fig. 14– LSTM chain.

The calculations that allow to solve the LSTM paradigm are [16]:

$$
\begin{aligned}
f_t &= \sigma_g\left(W_f x_t + U_f h_{t-1} + b_f\right),\\
i_t &= \sigma_g\left(W_i x_t + U_i h_{t-1} + b_i\right),\\
o_t &= \sigma_g\left(W_o x_t + U_o h_{t-1} + b_o\right),\\
\tilde{c}_t &= \sigma_c\left(W_c x_t + U_c h_{t-1} + b_c\right),\\
c_t &= f_t \cdot c_{t-1} + i_t \cdot \tilde{c}_t,\\
h_t &= o_t \cdot \sigma_h\left(c_t\right).
\end{aligned}
\tag{6}
$$

We detail below the information in the diagram:

Known data:

- $x_t \in R^d$ – input vector to the LSTM unit,

d and h - number of input features and number of hidden units, respectively.

Unknowns:

- $f_t \in R^h$ – forget gate's activation vector,
- $i_t \in R^h$ – input/update gate's activation vector,
- $o_t \in R^h$ – output gate's activation vector,
- $h_t \in R^h$ – hidden state vector (output vector of the LSTM unit),
- $c_t \in R^h$ – cell input activation vector,
- $c_t \in R^h$ – cell state vector,
- $W \in R^{hxd}$, $U \in R^{hxd}$, $b \in R^h$ – weight matrices and bias vector parameters which need to be learned during training.

Activation functions are:

- σ_g – sigmoid function,
- σ_c – hyperbolic tangent function,
- σ_h – hyperbolic tangent function or, linear function.

3.2.2. LSTM training and classification

An optimization RNN training using LSTM units on data sequences is made in a supervised mode by a set of algorithms like gradient descent, combined with backpropagation through time to compute the gradients needed during the optimization process, in order to change each weight of the LSTM network in proportion to the derivative of the error (at the output layer of the LSTM network) with respect to corresponding weight. With LSTM units, when error values are backpropagated from the output layer, the error remains in the LSTM unit's cell. This "error carousel" continuously feeds error back to each of the LSTM unit's gates, until they learn to cut off the value. This allows to avoid the problem with standard RNNs where error gradients vanish exponentially with the size of the time lag between important events. This happens for sub unitary norm of the weights. The system is trained using the equations (6).

4. EXPERIMENTAL RESULTS

The experiments aimed at evaluation of the FFNN and LSTM in the context of three classes of sounds: chainsaw, genuine forest and vehicles. The two neural networks will be run on the two types of feature spaces: the mel-cepstral features,

accompanied or not by ZCR and Spectral Flatness. and the Fourier power spectrum spaces. We will compare the obtained results with the former performance obtained using GMM and DTW. The experiments were closed set identification and performance given by identification rates.

At the framing layer we have evaluated segments of 3 s. We have assessed several lengths of subframes (analysis frames), based on the above remark (see Fig. 2–5). So, the lengths considered are mainly 22 ms, 44 ms, 88 ms. Concerning the frequency interval length, we have investigated lengths of 3.7, 7.4, 10 and 12 kHz.

The acoustic material contains 99 recordings of the three classes of sounds, 39 were used for training and 60 for testing, the testing set resulted in 685 segments of three seconds.

4.1. EXPERIMENTS USING THE FFNN

We have applied FFNN methodology feeding at input Mel-cepstral features (coming with or without Spectral Flatness and ZCR) and Fourier spectrum features. We have extracted 12 to 20 Mel-cepstral coefficients on different frequency intervals. At training we fed the information at the of sample layer, each sample being associated with the expected outputs 1, 0, –1, depending on the nature of the sound sample (chainsaw, genuine forest, vehicle engines). A sample in this case means a feature vector (of Mel-cepstral coefficients or Fourier coefficients, etc.). Another approach would have been to train the network at the layer of each segment, by associating the above expected output values depending on the nature of the segment. We applied the batch training and used the MATLAB framework. We evaluated the Bayesian Regularization, Levenberg-Marquardt Algorithm (LMA) as training solutions. At classification we evaluated each segment of 3 s, by assessing each of its samples. A sample belongs to a certain class if its output score is closest to that class expected output. The overall decision on the 3 s layer is taken by applying one of the rules:

- Majority voting (the segment is associated with the class for which most of the samples of the segment belong to the respective class)
- Average output: the average output score of the samples on the segment is closest to the expected output of a certain class.

Concerning the network architecture, we have tested FFNN with 1 to 4 layers, with 6 to 10 neurons on each layer.

4.1.1. Experimental results

As the performance of the test depends on the initialization in the training ser we provided 5 tests for each configuration. The tables below present some relevant results.

Table 1

FFNN – MFCC results of 5 tests using networks of 3 layers, with 8, 7, 6 neurons respectively 88ms analysis windows, 0–10 kHz, 17 Mel-coefficients, and Spectral flatness. Training accomplished by Bayesian Regularization, and classification using the majority voting rule

	Chainsaw	*forest*	*vehicle*	*General*
test1	51.30	95.69	61.53	**70.27**
test2	53.40	79.31	58.84	**64.27**
test3	46.07	92.67	56.53	**65.88**
test4	49.73	93.96	57.30	**67.64**
test5	53.40	94.82	57.69	**69.10**

Table 2

FFNN – Power Spectrum results of 5 tests using networks of 4 layers, with 9, 8, 7, 6 neurons respectively, 88ms analysis windows, 0–7400 Hz, training models obtained by Bayesian Regularization and classification using the majority voting rule

	Chain saw	*forest*	*vehicle*	*General*
test1	74.86	92.24	74.61	**80.67**
test2	65.96	91.37	78.46	**79.35**
test3	78.01	87.06	77.69	**80.96**
test4	67.53	80.17	78.46	**75.98**
test5	75.39	77.58	78.07	**77.16**

Table 3

FFNN – Power Spectrum – results of 5 tests using networks of 3 layers, with 9, 8, 7 neurons respectively, 88ms analysis windows, 0–3700 Hz, training models obtained by LMA and classification using the majority voting rule

	Chain saw	*forest*	*vehicle*	*General*
test1	72.77	84.48	77.69	**78.62**
test2	75.39	79.31	85.76	**80.67**
test3	69.11	89.65	69.23	**76.13**
test4	69/63	90.08	79.61	**80.38**
test5	64.92	87.93	84.23	**80.08**

Table 4

FFNN – Power Spectrum – results of 5 tests using networks of 3 layers, with 10, 9, 8 neurons respectively, 88ms analysis windows, 0–7400 Hz, training using LMA and classification using the average score on the 3ms frames

	Chain saw	*forest*	*vehicle*	*General*
test1	52.36	98.28	67.69	**73.79**
test2	54.45	97.41	65.39	**73.21**
test3	52.88	97.41	64.62	**72.47**
test4	57.07	93.10	64.62	**72.18**
test5	59.16	86.21	68.08	**71.74**

Table 5

Average Identification rates obtained using FFN- DFT with 3 layers with sizes 9,8, 7, using different values for f_{high} and lengths of the analysis frame, training with Bayesian Regularization and majority vote classification

		analysis frame lengths		
		22	44	88
Frequency interval	3700	71.04	76.31	78.83
	7400	74.98	76.02	77.22
	10000	69.64	74.76	76.02
	12000	72.61	75.17	74.00

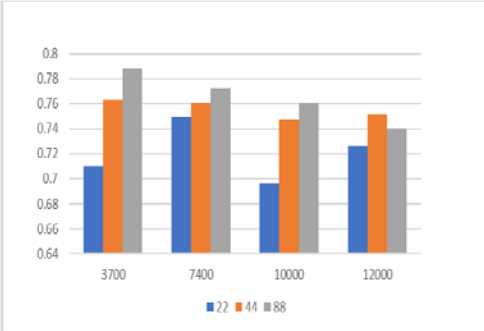

Fig. 15. Average Identificarion rates for different values of f_{high} and length of the analysis frame

Table 1 shows maybe the best performance obtained using Mel-cepstral features as input, using a 3 layers FFNN, with 8, 7, 6 neurons on each layer, 88ms analysis windows, 0–10 kHz, frequency interval for which the coefficients were computed. 17 Mel-coefficients were extracted, and Spectral flatness added on each analysis frame. Training was accomplished by Bayesian Regularization, and classification using the majority voting rule. The average performance was 67.43%. This performance is lower than, or comparable to the one obtained previously applying the classical GMM and DTW approaches [11]. Table 2 presents the results of 5 tests using FFNN with Power Spectrum coefficients as input, of 4 layers, with 9, 8, 7, 6 neurons respectively, 88 ms analysis windows, 0–7400 Hz frequency interval for spectral features. At training we applied Bayesian Regularization and at classification the majority voting rule. The average performance was 78.82%. Table 3 presents the results of 5 tests using networks of 3 layers, with 9, 8, 7 neurons respectively, 88 ms analysis windows, 0–3700 Hz, applying LMA training and classification using the majority voting rule. The average recognition rate was 79.17%. Table 4 contains the results of 5 tests applying FFNN to spectral coefficients, calculated on the frequency domain 0–7400 Hz and 88 ms analysis windows, using the LMA training, on 3 layer networks with 10, 9, 8 neurons. The classification algorithm used the average score on 3 s frames. The average performance was 72.67%.

Discussing the overall results, the Fourier spectrum as input to FFNN yielded very good results when applying the classification majority vote rule. The average score rule produced poorer results but still are better than using Mel-cepstral analysis or GMM, DTW [11]. The Bayesian Regularization and LMA produced comparable results, maybe LMA results were more balanced among the 5 tests. Concerning the network architecture for the Fourier spectrum variants of 2, 3 or 4 layers produce comparable results, especially when using the majority voting rule. For the average score classification, the 3 layers FFNN seemed to work better than 4 layers nets. Concerning the analysis window, the results are better in all the cases for lengths of 44ms or 88ms. Table 5 and Fig. 15 present the average overall identification rates for the FFNN-DFT approach using Bayesian regulation at training, and majority voting at classification, several analysis window lengths and frequency intervals. The best average score is obtained for the spectrum restricted to 0–3700 Hz, and an analysis window of 88 ms, but in fact the results are very close among the frequency intervals and among the 5 tests for each configuration there were many identification rates above 80%.

4.2. EXPERIMENTS USING THE LSTM

In the experiments using LSTM we used the same input as in the FFNN experiments. The number of hidden units was set to 100 and each cell configured with 5 layers, the default MATLAB configuration.

4.2.1. Experimental results

Table 6 presents the best results obtained so far by applying LSTM. We have fed as input 18 dimensional sheer Mel-cepstral vectors, calculated on 44ms analysis window, and filtering the frequency domain to 0–12 kHz. The average performance among the 5 tests is 64.85%. As can be seen the identification rates are unbalanced among the three classes. In any other configurations the results were even worse.

In what concerns the experiments using as input the Fourier spectrum we failed to obtain interesting results, as the network did not behave well from the beginning at training.

Table 6

Results of 5 tests using LSTM applied on an input set of Mel-cepstral 18-dimensional vectors, calculated on 44ms analysis windows, and the frequency interval of [0, 12] kHz

	csaw	*forest*	*vehicle*	*g-ral*
test1	48.69	86.20	47.69	61.05
test2	37.69	93.10	56.92	63.83
test3	42.40	97.41	45	62.07
test4	67.53	90.51	47.69	67.78
test5	62.82	93.10	53.46	69.54

Fig. 16 – Accuracy estimation during training for a LSTM-MFCC process.

Fig. 17 – Accuracy estimation during training for a LSTM applied on Fourier power spectra.

Figures. 16, 17 present the estimation of the achieved accuracy during the training process for LSTM applied to Mel-cepstral input and power spectra respectively. While the first process achieve maximum accuracy in less than 100 iterations the LSTM applied to power spectra aschieves less than 80% in more than 300 iterations.

5. CONCLUSIONS AND FUTURE WORK

We have designed and laid out a framework based on Deep Neural networks and aimed at some problems of environmental sound recognition. We used two types of networks (Deep Feedforward Neural Network and LSTM) and have fed as inputs two types of data, Mel-cepstral and Fourier power spectral coefficients. Deep Feed Forward Neural Network output the best results, mainly when using the sheer spectral features. and especially when using the majority voting rule, with an average identification rate of over 78%, with about 10% higher than other methods performance. This fact suggests that FFNN, based on Fourier spectral features, using a less complex processing sequence, is able to produce more valuable features than the elaborate Mel cepstral analysis. A difference is in the number of features at input, while the Mel features are fewer than 20, the spectrum on 0–7400 Hz frequency interval means about 170 coefficients. Figures. 18, 19 summarize this idea.

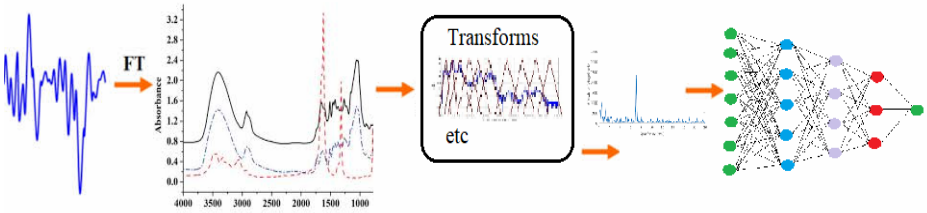

Fig. 18 FFNN using Mel cepstral input applies a range of transforms on the Fourier spectrum and feeds the result to the network.

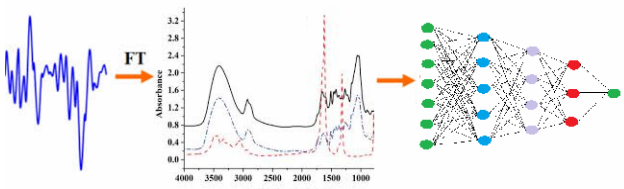

Fig. 19 FFNN using Fourier Spectrum coefficients as has a simpler schema, and probably devises more valuable features through the layers of the network.

The disappointing results using the LSTM network may have several reasons. One of them is maybe the unproper use of the LSTM MATLAB tool. A second reason may be in the fact that this type of network might be not suited to the kind of problem we want to solve.

Another advantage of using FFNN is the fact that it is easy to implement in programming environments other than MATLAB. While the models can be generated in MATLAB, the classification part can be implemented in more inexpensive programming languages, like C++, Java, etc., using the parameters established at training.

As future work we intend to resume the tests with LSTM approach, on one side and with other DNN variants. Concerning future developments with FFNN we would like to continue the research by placing the training and classification, not at the sample layer but at 3s segment layer.

Another important objective is to extend the field of research to other AESR applications, in the field of scientific environment monitoring (e.g. detect bird or species), or early detection of disasters such as land sliding or avalanches, where acoustic emissions are among the data used as input.

Acknowledgements. This work has been supported in part by UEFISCDI Romania through projects SeaForest, CitiSim, SenSyStar and WINS@HI, and funded in part by European Union's Horizon 2020 research and innovation program under grant agreement No. 777996 (SealedGRID project) and No. 643963 (SWITCH project).

Received on October 19, 2020

REFERENCES

1. HUANG, J., OHNISHI, N., SUGIE, N., *Building ears for rbotos: Sound localization and separation*, Artificial Life and Robotics, **1**, *4*, pp. 157–163, 1997.
2. BABIŠ, M., ĎURÍČEK, M., HARVANOVÁ, V., VOJTKO, M., *Forest Guardian –Monitoring System for Detecting Logging Activities Based on Sound Recognition*, IIT.SRC, pp. 1–6, 4 May, 2011.
3. PAPÁN, J., JURECKA, M., PÚCHYOVÁ, J., *WSN for Forest Monitoring to Prevent Illegal Logging*, IEEE, Proceedings of the Federated Conference on Computer Science and Information Systems, pp. 809–812, 2012.
4. VENIER, L. A., MAZEROLLE, M. J., RODGERS, A., MCILWRICK, K. A., HOLMES, S., THOMPSON, D., *Comparison of semiautomated bird song recognition with manual detection of recorded bird song sample*, Avian Conservation and Ecology **12**, *2*, 2017, https://doi.org/10.5751/ ACE-01029-120202.
5. MITROVIC, D., ZEPPELZAUER, M., BREITENEDER, C., *Discrimination and Retrieval of Animal Sounds*, Poster: IEEE Multimedia Modelling Conference, Beijing; 01-03-2006 - 01-06-2006; in: Proceedings IEEE Multimedia Modeling Conference, pp. 339–343, 2006.
6. COWLING, M., SITTE, R., WYSOCK, T., *Analysis of Speech Recognition Techniques for use in a Non-Speech Sound Recognition System*, Advanced Signal Processing for Communication Systems, Springer, 2002.

7. COWLING, M., SITTE, R., *Recognition of Environmental Sounds Using Speech Recognition Techniques*, In book: Advanced Signal Processing for Communication Systems, April 2006.
8. SACHIN CHACHADA, JAY KUO, C.-C, *Environmental Sound Recognition: A Survey*, APSIPA Transactions on Signal and Information Processing, **3**, 2014.
9. SIGTIA, S., STARK, A., KRSTULOVIC, S., PLUMBLEY, M., *Automatic environmental sound recognition: Performance versus computational cost*, IEEE/ACM Transactions on Audio, Speech, and Language Processing, **24**, *11*, 2016.
10. DAVIS, S. B. AND MERMELSTEIN, P., *Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences*, IEEE Trans. on ASSP, **28**, pp. 357–366, 1980.
11. OLTEANU, E., SEGĂRCEANU, S., GAVAT, I., *Evaluation of speech specific and non-speech classification techniques for environmental sound recognition*, SISOM 2019 (Symposium of Acoustics), Bucharest, May 2019.
12. https://en.wikipedia.org/wiki/Artificial_neural_network#/media/File:Neuron3.png.
13. BEALE, M. H., HAGAN, M. T., DEMUTH, H. B., *Neural Network Toolbox™ User's Guide*, edited The MathWorks, Inc., Natick, Mass., 2010, available at https://www2.cs.siu.edu/~rahimi/cs437/slides/nnet.pdf.
14. Zacharias, V., *AI for Data Science Artificial Intelligence Frameworks and Functionality for Deep Learning, Optimization, and Beyond*, Technics Publications, LLC, 2018.
15. https://en.wikipedia.org/wiki/Long_short-term_memory#/media/File:The_LSTM_cell.png.
16. SAK, H., SENIOR, A., AND BEAUFAYS, F. (2014a), *Long Short-Term Memory recurrent neural network architectures for large scale acoustic modeling*, In Proc. Interspeech.