# Learning Representations from Audio using Autoencoders

Nallaperumal.K
UG Scholar
PSG College of Technology

**Abstract:- Deep learning approaches allow us to look at signal processing problems from a different angle, which is currently widely disregarded in the music business. Audio is intrinsically more time-sensitive than film. You can never assume that a pixel in a spectrogram belongs to a single object. Due to the fact that audio is always transparent, spectrograms show all audible sounds overlapping in the same frame. It has been demonstrated that spectrograms can be processed as images and neural style transfer can be performed with CNNs, although the results have not been as exact as they have been for visual images. We should focus our efforts on developing more accurate representations.**

***Keywords**:- Autoencoders, Autocorrelogram, Encoding, Audio Encoders, RNN Autoencoder, Audio Frequency, Auto Correlation And Convolution, Cross Fold Validation.*

## I. INTRODUCTION

In today's world, time and accuracy are major concerns. Any product that can effectively reduce time and provide result with better accuracy is accepted and appreciated. In earlier days, when audios are encoded with various methods, they either results in loss of data or temporal inaccuracies. Thus, we come up with providing better accuracy when reconstructing the encoded audio files. Here, we use autoencoder to learn a representation (encoding) for a set of data, typically for dimensionality reduction and use LSTM networks to classify, process and make predictions based on time series data. Earlier, existing systems represents auto correlogram in 2D view with power and time in its both axes, but here representing auto correlogram in 3D view which includes time, power, frequency in order to provide better accuracy. Many audio encoders are used to compress raw audio files. These encoders suffer from temporal inaccuracies and therefore cannot be used to reconstruct the same audio files. In this paper, a new system of audio encoding is proposed which includes compressing of temporal information and thereby minimizing data loss during compression.

To convert the audio from uncompressed format to a compressed format. Encoding makes audio signal into system readable format. It also reduces space and dimensionality. This system can help to replace the existing technologies where audio compression is treated as major part. Audio data should be efficiently encoded into compressed format using RNN autoencoder by interacting with the knowledge/data store. Thus, the result of the compressed format should be an accurate representation of audio waves. Then, audio waves are being reconstructed into audio format without much loss of original data.

## II. ANALYSIS AND DESIGN

### 2.1 ANALYSIS:

### 2.1.1 EXISTING SYSTEM:

In existing system, audio file is given to auto correlogram which provides 2D audio wave which has power and time in its two axes, then 2D view of audio files are directly fed into ANN encoder which resulted in lesser accuracy of compressed file. ANN encoder can accept only single input file at a time. ANN encoder is better suited for text files. But, when on using ANN encoders for audio files, the accuracy reduced drastically on comparison with text files.

### 2.1.2 PROPOSED SYSTEM:

In this system, audio files are first fed into auto correlogram which provides three dimensional view of audio wave which includes power, time, frequency. Then, the output is given to RNN encoder which performs sequential input and output operation to provide better accuracy. This process is repeated until maximum accuracy gets obtained. This encoder allows a time sequence to have temporal dynamic behaviour.

### 2.1.3 ADVANTAGES:

1. Better noise free audio file.
2. Three dimensional autocorrelogram helps in providing better accuracy.
3. RNN encoder performs sequential input and output operation.
4. Better temporal dynamic behaviour.

## 2.2 DESIGN:

### 2.2.1 ARCHITECTURE DIAGRAM

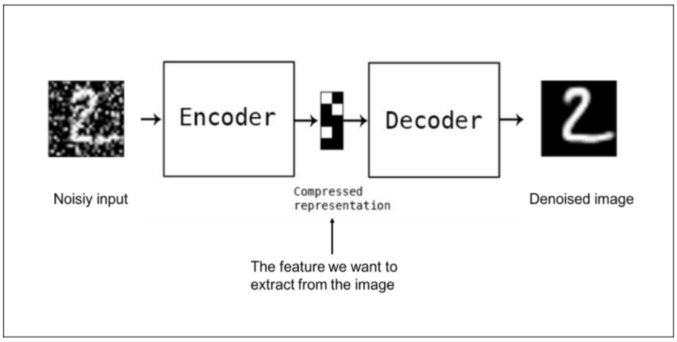

Encoder
Decoder
Noisiy input
Compressed representation
Denoised image
The feature we want to extract from the image

Figure 1: architecture diagram

1 Consider an audio file.
2 Input the audio file into auto correlogram generator through audio inputter, it will generate an auto correlogram as output.
3 The auto correlogram will give 3-dimensional graph as output.
4 The auto correlogram thus generated will be given as input to RNN encoder.
5 The RNN encoder will interact with existing datastore and analyse the auto correlogram with the data present in the dataset.
6 Now, compressed audio file has generated.
7 On decoding the compressed audio file, original audio file will be generated which is a noise free one.

### 2.2.2 COMPONENT DIAGRAM

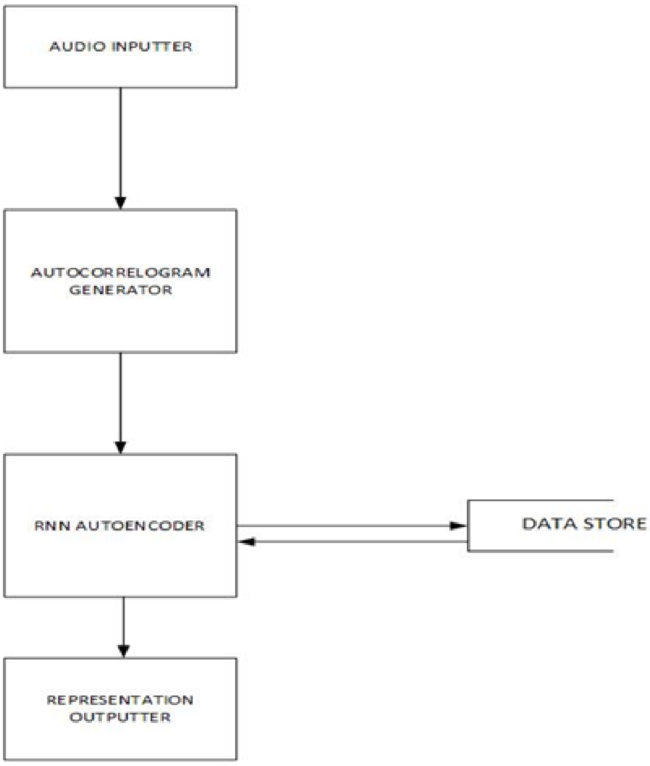


Figure 2: component diagram

The audio inputter is a separate component that interacts with the user. The auto correlogram generator gets audio file as input and generates auto correlogram as output. The RNN autoencoder gets the output of auto correlogram as input and also interacts with the knowledge/data store to provides the final representation. The hidden layer is taken as result from autoencoder as learned representation.

### 2.2.3 ACTIVITY DIAGRAM

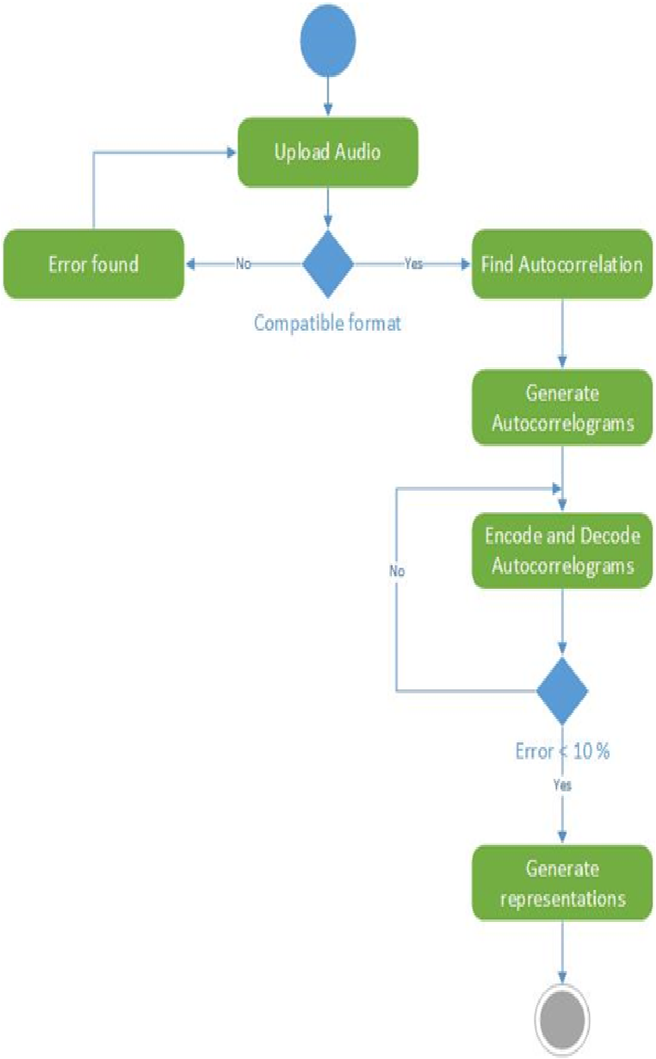


Figure 3: activity diagram

1. Insert a file into system, if it is an audio file system accepts the file and directs to autocorrelation, else error occurs because file is in incompatible format.
2. The output wave of autocorrelation is given as input to auto correlogram and it will generate a three dimensional wave.
3. Now, the obtained wave is fed into RNN autoencoder and it generates a compressed file which will be in encoded format.
4. And this can be decoded to get back the original audio file.
5. This process is repeated until the error can be minimized to maximum extent.

### 2.2.4 USECASE DIAGRAM

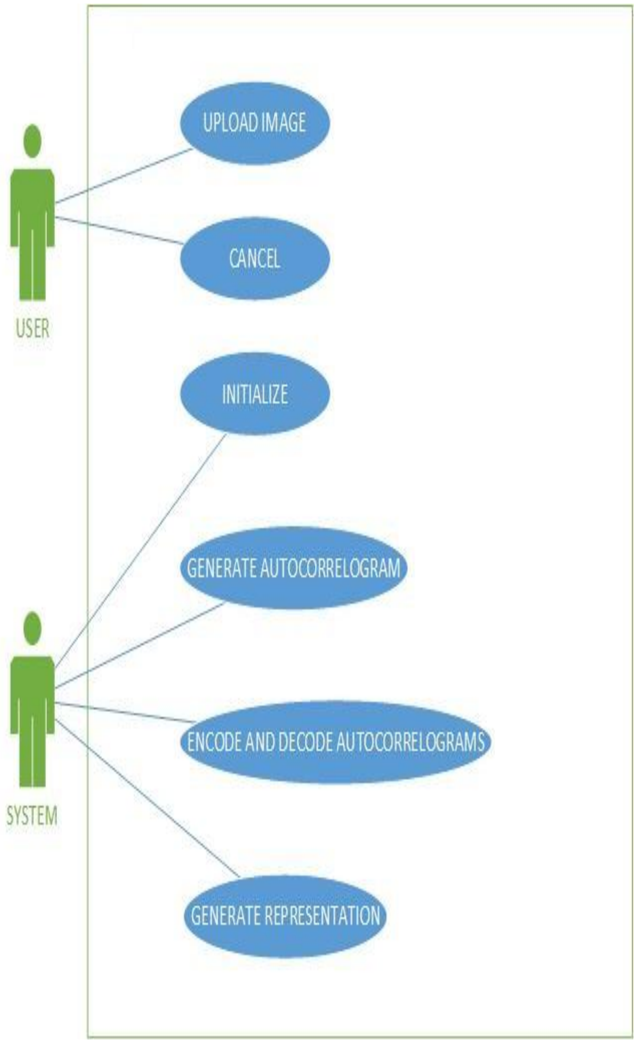


Figure 4: use case diagram

**USER:**

1. User can upload an audio file
2. User can cancel the ongoing encoding process

**SYSTEM:**

1. Initially it checks for compatible format.
2. It gets audio file as input to auto correlogram and it sends the output of this process to encoder.
3. Then system decodes the encoded audio file.
4. Repeat this process until better accuracy is obtained.

## III. IMPLEMENTATION

### 3.1 System Implementation:

### 3.1.1 FFT Algorithm

The time and frequency domains in complex notation each contain one signal made up of N complex points. The real component and the imaginary part of each of these complex points are made up of two numbers. When we talk about complicated sample X, for example, we're referring to the combination of ReX and ImX. To put it another way, each complex variable has two numbers. When two complex variables are multiplied, the four distinct components must be added together to generate the product's two components. Signal, point, sample, and value are singular terms that relate to the mixture of the real and imaginary parts.

Steps:

- The FFT works by dividing an N-point time domain signal into N time domain signals, each of which consists of a single point.
- The N frequency spectra corresponding to these N time domain signals are calculated in the second step. Finally, the N spectra are combined to form a single frequency spectrum. A bit reversal sorting technique is typically used to decompose the FFT time domain. This entails counting in binary with the bits flipped left-to-right to rearrange the order of the N time domain samples.
- The next step within the FFT rule set is to discover the frequency spectra of the warnings in the 1-factor time domain. Each of the 1-factor alarms is now a frequency spectrum and no longer a signal in the time domain**.**
- The last step in the FFT is to combine the N frequency spectra in exactly the reverse order in which the time domain decomposition took place.

### 3.1.1.1 FUNDAMENTAL FREQUENCY

To identify the period $T$ , the frequency $f = 1/T$ , or the angular frequency $\omega = 2\pi f = 2\pi/T$ of a given sinusoidal or complex exponential signal, it is always helpful to write it in any of the following forms:

$$\sin(\omega t) = \sin(2\pi f t) = \sin(2\pi t/T)$$

The fundamental frequency of a signal is the greatest common divisor (GCD) of all the frequency components contained in a signal, and, equivalently, the fundamental period is the least common multiple (LCM) of all individual periods of the components.

### 3.1.2 RECURRENT NEUTRAL NETWORK

A recurrent neural network (RNN) is also a class of artificial neural network where connections between nodes are somewhat like a directed graph on a sequence. This enables it to exhibit temporal dynamic behaviour for a time sequence. Unlike feedforward neural networks, RNNs will use their internal state to method sequences of inputs. This makes them more appropriate for tasks like non segmental, connected handwriting recognition or speech recognition.

The term RNN is employed to check with 2 broad categories of networks with the same general structure, wherever one is finite impulse and therefore the alternative is infinite impulse. Each categories of networks exhibit temporal dynamic behaviour. A finite impulse of continual network may be a directed acyclic graph that may be unrolled and replaced with a strictly feedforward neural network, whereas associate in Nursing infinite impulse continual network may be a directed cyclic graph that can't be unrolled.

Both infinite and finite impulse continual networks have extra keep state, and hence the neural network directly manages the storage. Another network or graph may also replace the storage, if in case that comes with has feedback loops or time delays. Such controlled states are cited as gated state or memory, and are counted as a part of long remembering networks (LSTMs) and gated continual units.

### 3.1.3 LSTM:

Units of a RNN network are short-term memory units. An LSTM network is an RNN that is comprised of LSTM units. An input and an output gate, a cell and a forget gate are combined to form a typical LSTM unit. The information flow into the cell and out of the cell is controlled by these three gates, and the values across arbitrary time intervals is remembered by the cell.

Because there might be lags of undetermined duration between critical occurrences in a time series, LSTM networks are well-suited to categorising, processing, and making predictions based on time series data. LSTMs were created to solve the problems of exploding and vanishing gradients that can occur when training regular RNNs**.** In many cases, LSTM has an advantage over RNNs, hidden Markov models, and other sequence learning approaches due to its relative insensitivity to gap length.

## IV. AUTOCORRELATION

The correlation of a signal with a delayed replica of itself as a function of delay is known as autocorrelation, sometimes known as serial correlation. It's the similarity of measurements as a function of the time lag between them, to put it another way. Autocorrelation analysis is a mathematical tool for detecting repeating patterns, such as the presence of a periodic signal disguised by noise or the identification of a signal's missing fundamental frequency inferred by its harmonic frequencies. It's frequently used in signal processing to examine functions or sequences of values, such as time domain signals.

The autocorrelation of a true or complicated random method is that the Pearson correlation between values of the method at completely different times, as a function of the 2 times or of the time lag. Let $\{X_t\}$ be a random process, and $t$ be any point in time ($t$ may be an integer for a discrete-time process or a real number for a continuous-time process). Then $\{X_t\}$ is the realization (or value) produced by a particular run of the process at time . Suppose that the process has mean $\mu_t$ and variance $\sigma_t^2$ at time , foreach $t$. Then the definition of the **auto-correlation function** between times $t_1$ and $t_2$ is

$$R_{XX}(t_1, t_2) = E[X_{t_1} X_{t_2}{}']$$

where $E$ is the expected value operator. Subtraction of the mean before multiplication yields the **auto-covariance function** between times $t_1$ and $t_2$ :

$$\begin{aligned} K_{XX}(t_1, t_2) &= E\left[(X_{t_1} - \mu_{t_1})(X_{t_2} - \mu_{t_2})'\right] \\ &= E\left[X_{t_1} X_{t_2}{}'\right] - \mu_{t_1}\mu_{t_2}{}' \end{aligned}$$

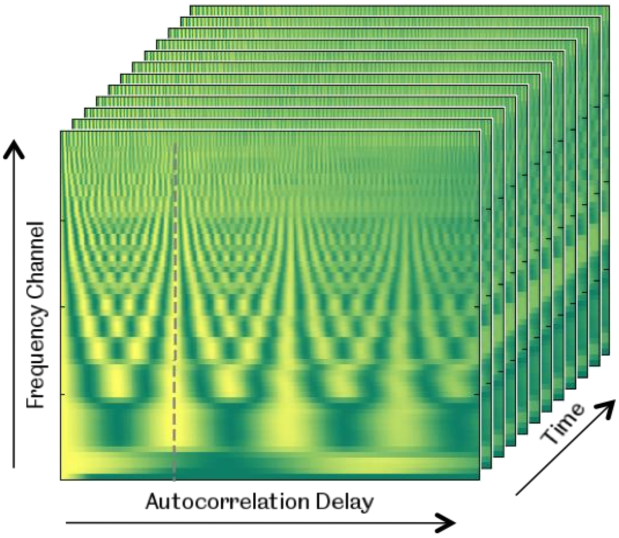


Figure 5: autocorrelogram

### 4.1. CONVOLUTIONS

A convolution is a mathematical integration function that indicates the degree of overlap between two functions when one is shifted over another. A convolution, on the surface, appears to be a blender that blends one function with another to minimise data space while keeping information.

In terms of Deep Learning and Neural Networks:

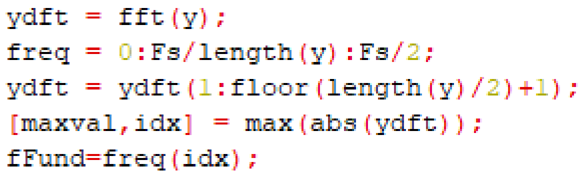
- Convolutions are learnable-parameter filters (matrices or vectors) that extract low-dimensional features from input data.
- They have the ability to keep spatial or positional links between input data points intact.
- By enforcing a local connection pattern between neurons in adjacent layers, convolutional neural networks take advantage of spatially local correlation.

Convolution, in its most basic form, is the process of applying the sliding window concept (a filter with learnable weights) to the input and creating a weighted sum (of weights and input) as the output. The feature space is the weighted sum, which is utilised as the input for the subsequent layers.

### 4.2 Signal Processing:

#### 4.2.1 Fundamental Frequency:

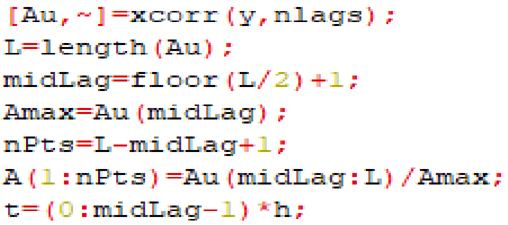
The fundamental frequency is calculated with the help of fast fourier transform and the frequency where the fft is at is maximum is chosen as fundamental frequency. Sometimes the maximum occurs at zero frequency where detrend (normalisation) is performed before fft.



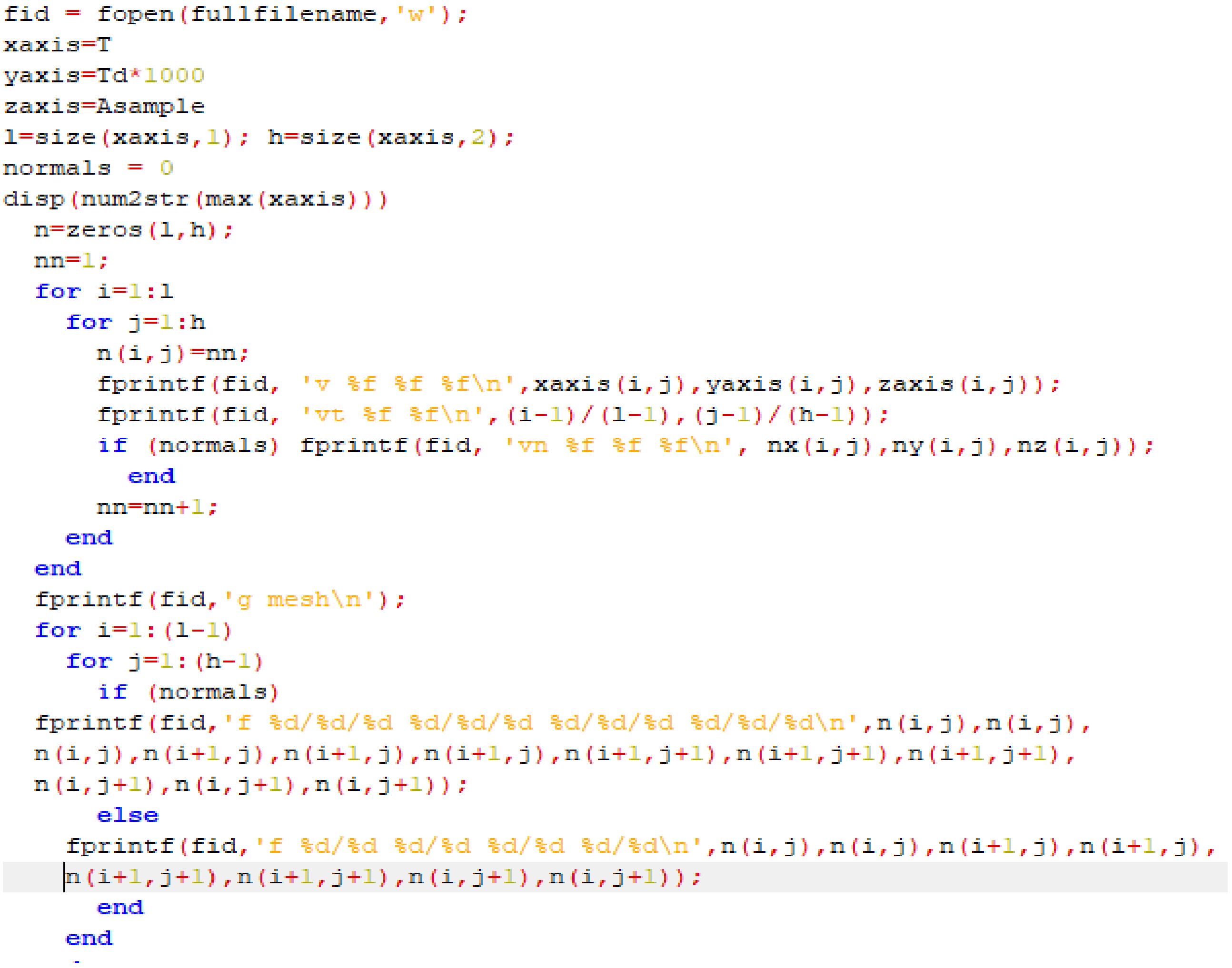
```
ydft = fft(y);
freq = 0:Fs/length(y):Fs/2;
ydft = ydft(1:floor(length(y)/2)+1);
[maxval,idx] = max(abs(ydft));
fFund=freq(idx);
```

### 4.2.2 Correlation

xcorr returns a 2*M-1 cross-correlation sequence for a vector length M from lags [-M:M] with the zeroth-lag in the middle of the returned vector. The auto-correlation (0-lag) value is going to be the max() so we restrict the search to lags>=1 and look for the maximum away from the initial decaying self-correlation around zero that can be significant depending on the shape of the two signals.



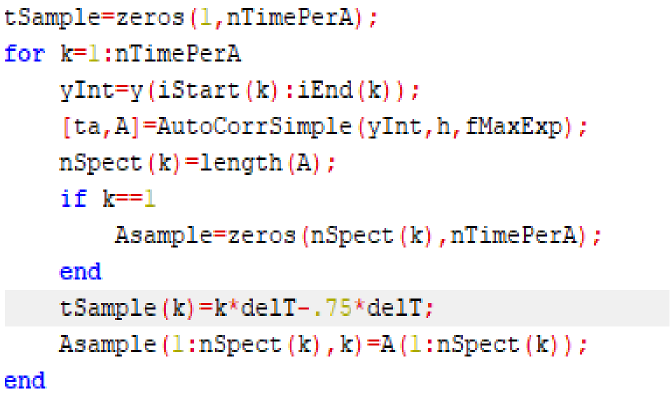
```
[Au,~]=xcorr(y,nlags);
L=length(Au);
midLag=floor(L/2)+1;
Amax=Au(midLag);
nPts=L-midLag+1;
A(1:nPts)=Au(midLag:L)/Amax;
t=(0:midLag-1)*h;
```



```
fid = fopen(fullfilename,'w');
xaxis=T
yaxis=Td*1000
zaxis=Asample
l=size(xaxis,1); h=size(xaxis,2);
normals = 0
disp(num2str(max(xaxis)))
  n=zeros(l,h);
  nn=1;
  for i=1:l
    for j=1:h
      n(i,j)=nn;
      fprintf(fid, 'v %f %f %f\n',xaxis(i,j),yaxis(i,j),zaxis(i,j));
      fprintf(fid, 'vt %f %f\n',(i-1)/(l-1),(j-1)/(h-1));
      if (normals) fprintf(fid, 'vn %f %f %f\n', nx(i,j),ny(i,j),nz(i,j));
        end
      nn=nn+1;
    end
  end
  fprintf(fid,'g mesh\n');
  for i=1:(l-1)
    for j=1:(h-1)
      if (normals)
  fprintf(fid,'f %d/%d/%d %d/%d/%d %d/%d/%d %d/%d/%d\n',n(i,j),n(i,j),
  n(i,j),n(i+1,j),n(i+1,j),n(i+1,j),n(i+1,j+1),n(i+1,j+1),n(i+1,j+1),
  n(i,j+1),n(i,j+1),n(i,j+1));
      else
    fprintf(fid,'f %d/%d %d/%d %d/%d %d/%d\n',n(i,j),n(i,j),n(i+1,j),n(i+1,j),
    n(i+1,j+1),n(i+1,j+1),n(i,j+1),n(i,j+1));
      end
    end
```

### 4.2.3 Autocorrelation

Cut the specific signal into k parts, find autocorrelation for each part and merge them.



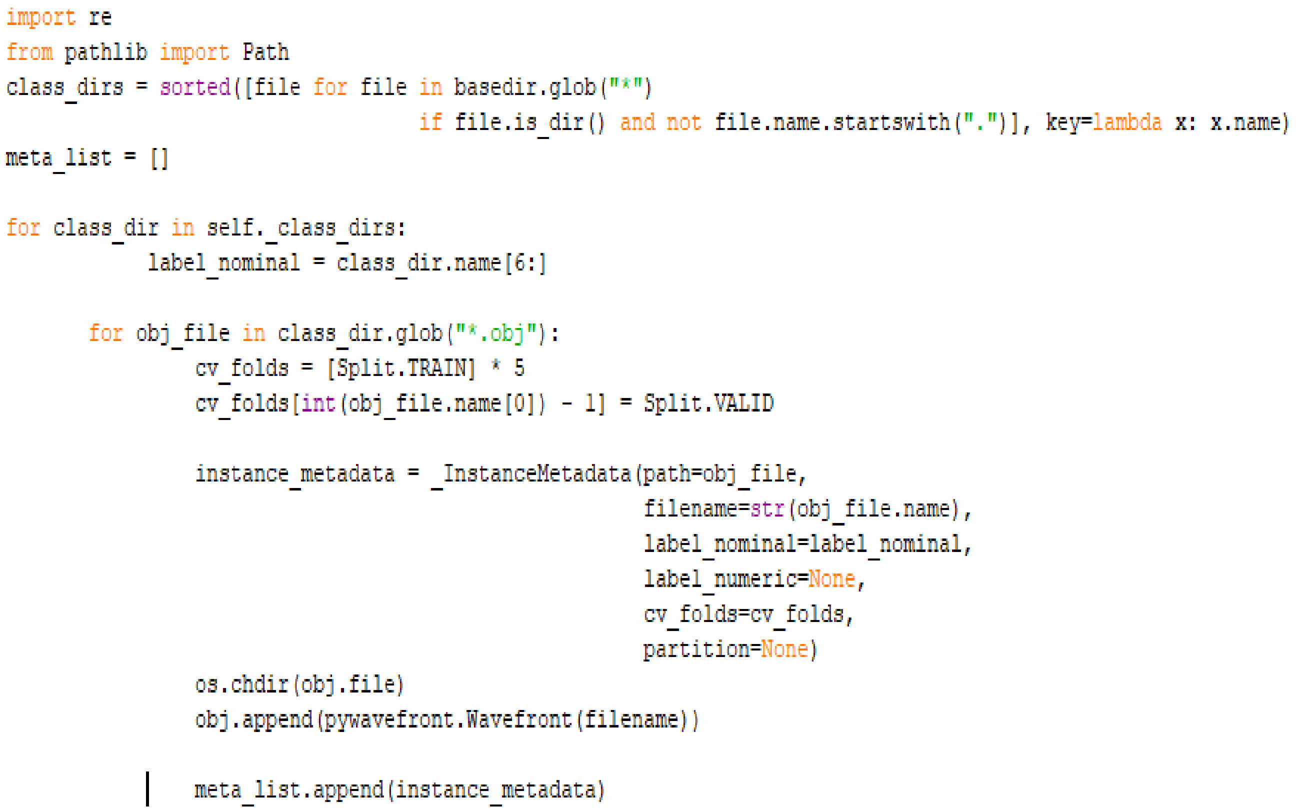
```
tSample=zeros(1,nTimePerA);
for k=1:nTimePerA
    yInt=y(iStart(k):iEnd(k));
    [ta,A]=AutoCorrSimple(yInt,h,fMaxExp);
    nSpect(k)=length(A);
    if k==1
        Asample=zeros(nSpect(k),nTimePerA);
    end
    tSample(k)=k*delT-.75*delT;
    Asample(1:nSpect(k),k)=A(1:nSpect(k));
end
```

### 4.2.4 Object Writer

We write the three dimensional data into the object files as time, frequency, and autocorrelation delay.

### 4.2.5 Object Parser

Pywavefront is used for reading the 3d object data.



```
import re
from pathlib import Path
class_dirs = sorted([file for file in basedir.glob("*")
                                   if file.is_dir() and not file.name.startswith(".")], key=lambda x: x.name)
meta_list = []

for class_dir in self._class_dirs:
            label_nominal = class_dir.name[6:]

       for obj_file in class_dir.glob("*.obj"):
                cv_folds = [Split.TRAIN] * 5
                cv_folds[int(obj_file.name[0]) - 1] = Split.VALID

                instance_metadata = _InstanceMetadata(path=obj_file,
                                                      filename=str(obj_file.name),
                                                      label_nominal=label_nominal,
                                                      label_numeric=None,
                                                      cv_folds=cv_folds,
                                                      partition=None)
                os.chdir(obj.file)
                obj.append(pywavefront.Wavefront(filename))

                meta_list.append(instance_metadata)
```

### 4.3 RNN Architecture

This architecture is made up of two models: one that reads the input sequence and encodes it into a fixed-length vector, and another that decodes the fixed-length vector and outputs the predicted sequence. Encoder-Decoder LSTM developed specifically for seq2seq situations is the name given to the architecture by combining the models. This LSTM is fully connected but the input and output layers may differ in applying activation functions as encoder introduces non-linearity while the decoder has a linear projection that helps in matching of output sequences with the input sequences.

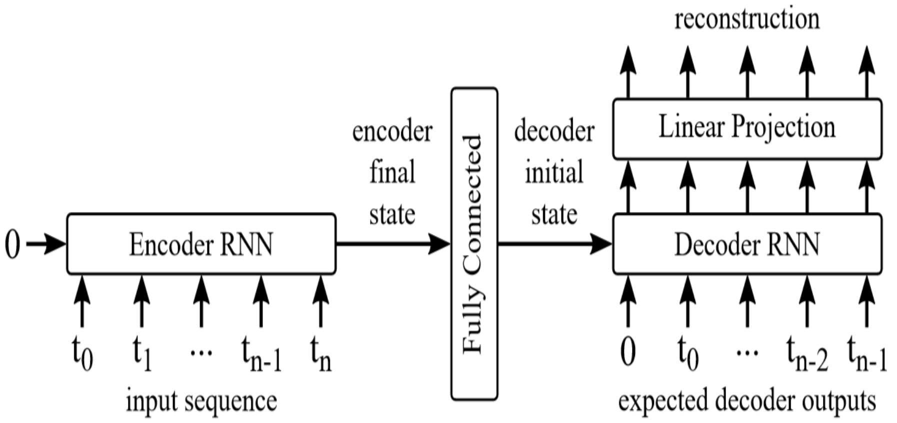


Figure 5: RNN Architecture

### 4.4 Dataset for ESC (Environmental Sound Classification)

The ESC-50 dataset consists of 2000 tagged environmental recordings equally balanced between fifty categories (40 clips per class). For convenience, they're sorted in five loosely defined major classes (10 per category):

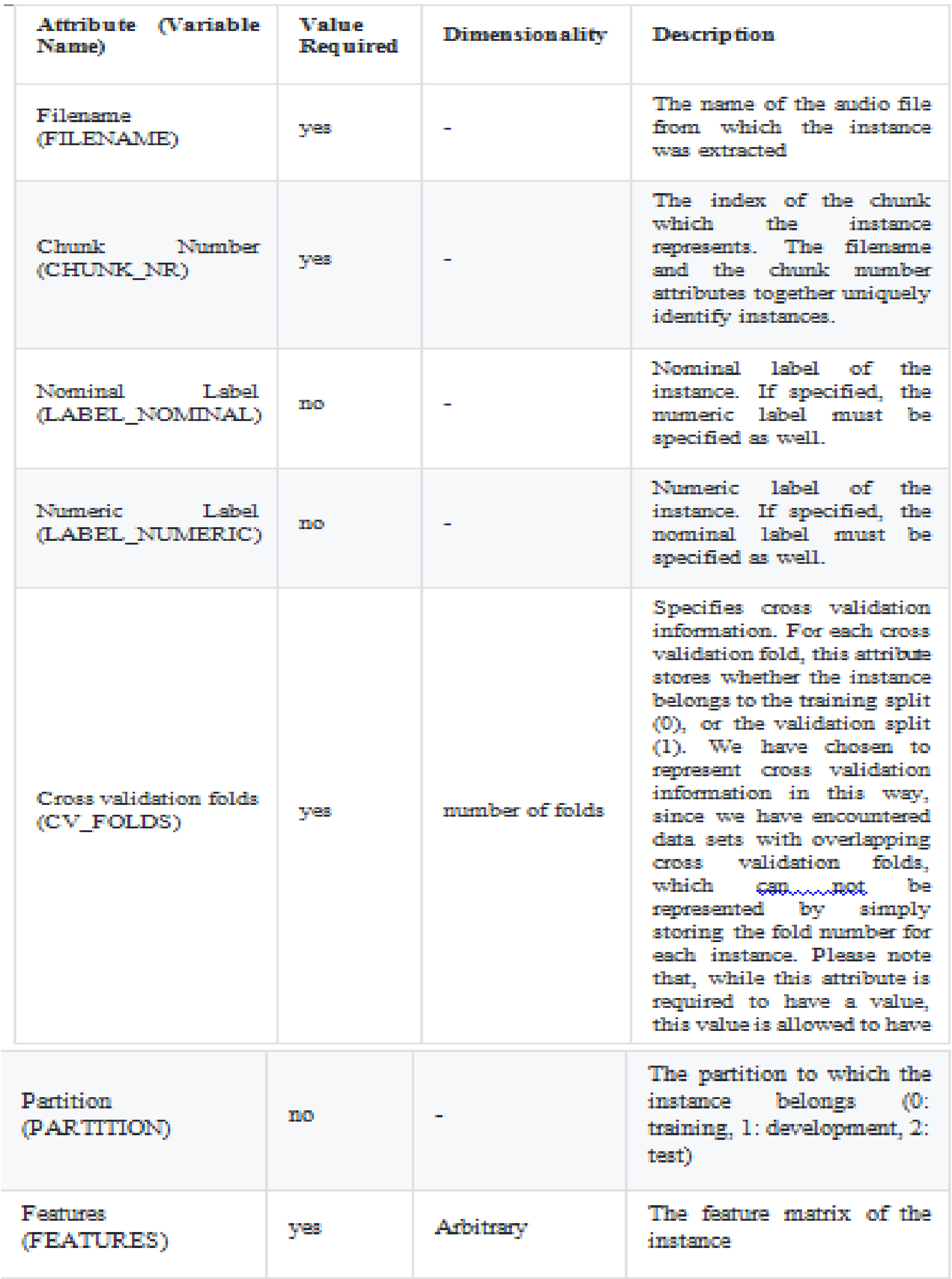
- natural soundscapes and water sounds,

- animal sounds,
- human (non-speech) sounds,
- exterior/urban noises.
- interior/domestic sounds,

When possible, the separation procedure aimed to hold on to sound occurrence in the foreground with little surrounding noise. Field recordings, on the other hand, are far from sterile, and some clips may still have audio overlap in the background. The dataset exposes you to a variation of sound sources, including those that are quite frequent (laughing, meow sounds by cat, barking of dogs), some that are quite different (glass breaking, cutting wood), and some noises that are more nuanced (helicopter and aeroplane noise).

### 4.5 Data Model Structure

Dataset store instances, which can correspond to either a whole audio file, or a chunk of audio files. For each instance, the attributes given below are stored.



| Attribute (Variable Name) | Value Required | Dimensionality | Description |
|---|---|---|---|
| Filename (FILENAME) | yes | - | The name of the audio file from which the instance was extracted |
| Chunk Number (CHUNK_NR) | yes | - | The index of the chunk which the instance represents. The filename and the chunk number attributes together uniquely identify instances. |
| Nominal Label (LABEL_NOMINAL) | no | - | Nominal label of the instance. If specified, the numeric label must be specified as well. |
| Numeric Label (LABEL_NUMERIC) | no | - | Numeric label of the instance. If specified, the nominal label must be specified as well. |
| Cross validation folds (CV_FOLDS) | yes | number of folds | Specifies cross validation information. For each cross validation fold, this attribute stores whether the instance belongs to the training split (0), or the validation split (1). We have chosen to represent cross validation information in this way, since we have encountered data sets with overlapping cross validation folds, which can not be represented by simply storing the fold number for each instance. Please note that, while this attribute is required to have a value, this value is allowed to have |
| Partition (PARTITION) | no | - | The partition to which the instance belongs (0: training, 1: development, 2: test) |
| Features (FEATURES) | yes | Arbitrary | The feature matrix of the instance |

Furthermore, we optionally store a label map, which specifies a mapping of nominal lab

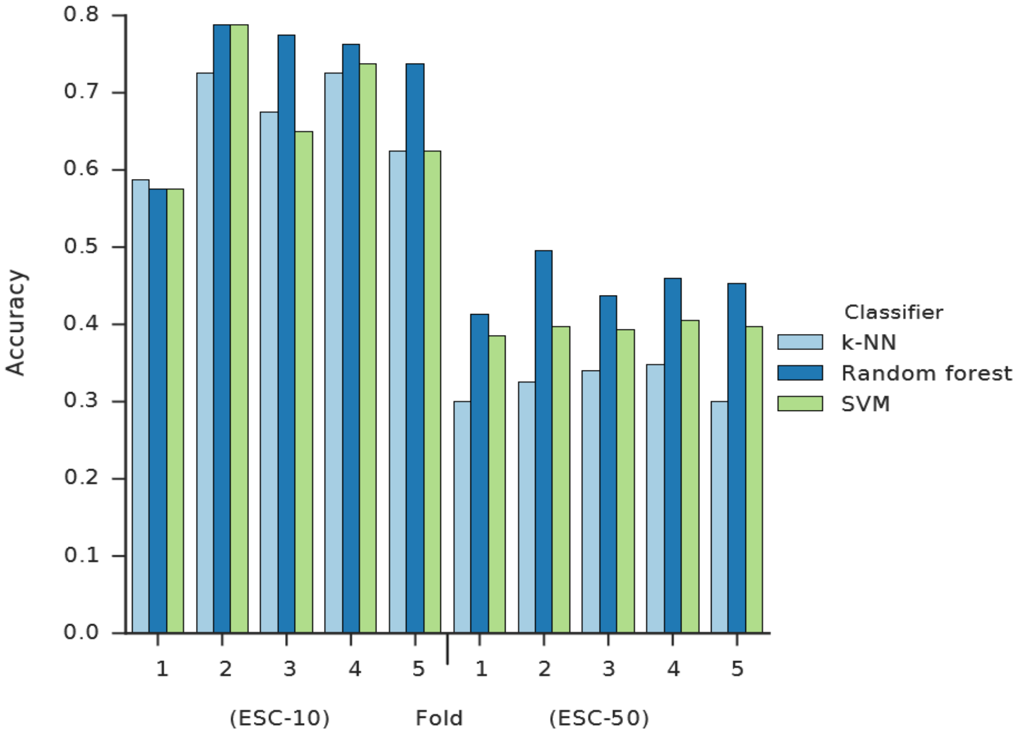
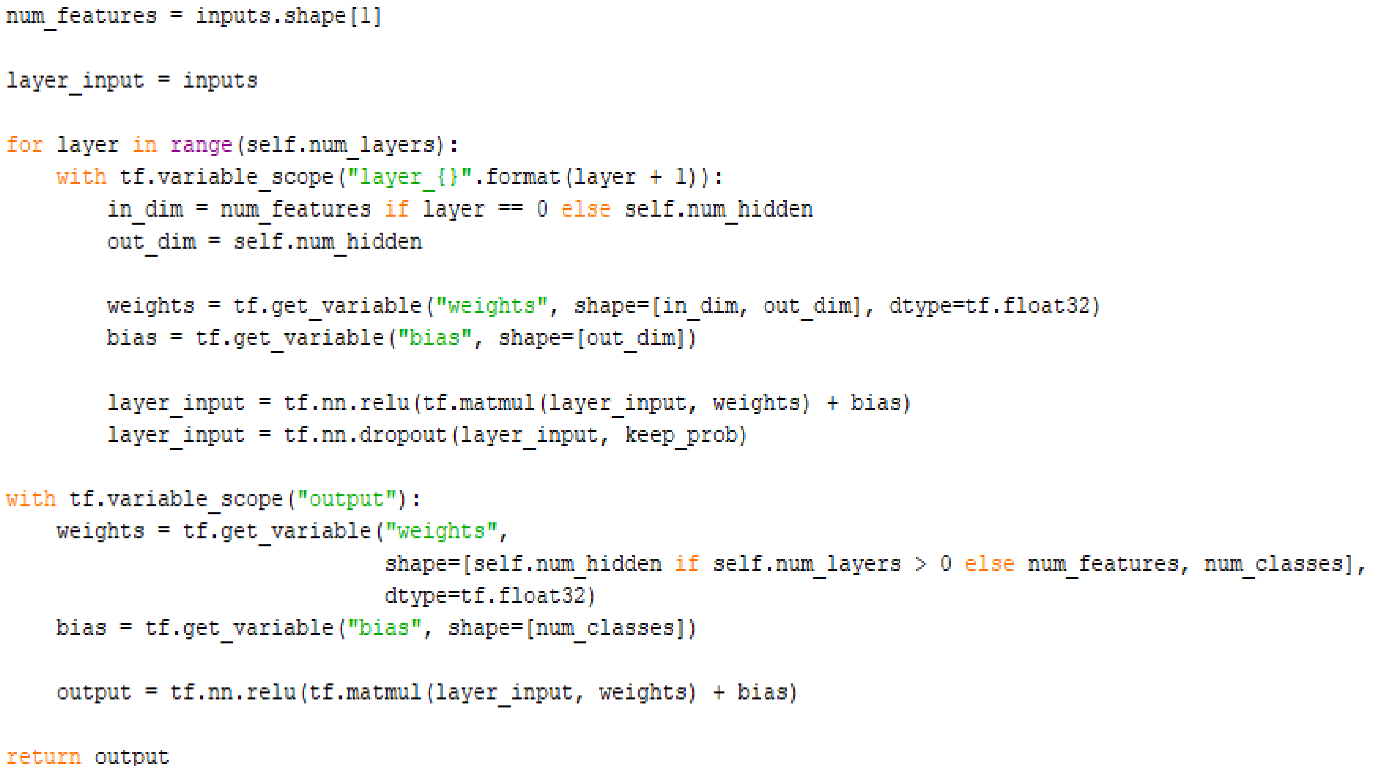


Figure 6: Data Model Structure

**Accuracy obtained from research paper,** ESC: Dataset for Environmental Sound Classification by, Karol J. Piczak Institute of Electronic Systems Warsaw University of Technology.

### 4.6 Multilayer Perceptron Classification:

We use a multilayer perceptron model to classify the learned representations from the audio data.



```
num_features = inputs.shape[1]

layer_input = inputs

for layer in range(self.num_layers):
    with tf.variable_scope("layer_{}".format(layer + 1)):
        in_dim = num_features if layer == 0 else self.num_hidden
        out_dim = self.num_hidden

        weights = tf.get_variable("weights", shape=[in_dim, out_dim], dtype=tf.float32)
        bias = tf.get_variable("bias", shape=[out_dim])

        layer_input = tf.nn.relu(tf.matmul(layer_input, weights) + bias)
        layer_input = tf.nn.dropout(layer_input, keep_prob)

with tf.variable_scope("output"):
    weights = tf.get_variable("weights",
                              shape=[self.num_hidden if self.num_layers > 0 else num_features, num_classes],
                              dtype=tf.float32)
    bias = tf.get_variable("bias", shape=[num_classes])

    output = tf.nn.relu(tf.matmul(layer_input, weights) + bias)

return output
```

## V. DEEP SPECTRUM

**Deep Spectrum** is a Python toolkit for feature extraction from audio data with pre-trained Image Convolutional Neural Networks (CNNs). It features an extraction pipeline which first creates visual representations for audio data - plots of spectrograms or chromagrams - and then feeds them to a pre-trained Image CNN. Activations of a specific layer then form the final feature vectors.
**(c) 2017-2018 Shahin Amiriparian, Maurice Gerczuk, Sandra Ottl, Björn Schuller: Universität Augsburg**

### 5.1 VISUALIZATION OF AUDIO WAVE

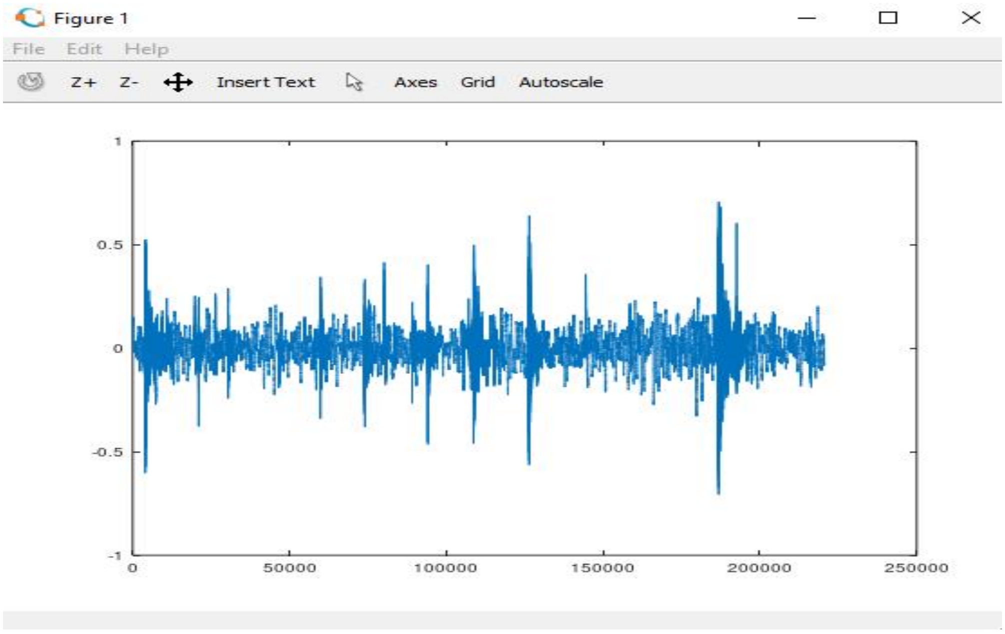


Figure 7: Audio Wave

The audio signal of a ogg file “1-4211-A.ogg” of class “Fire Cackling” is plotted to observe the amplitude and time(sec).

### 5.2VISUALIZATION OF 3D WAVE OBJECT

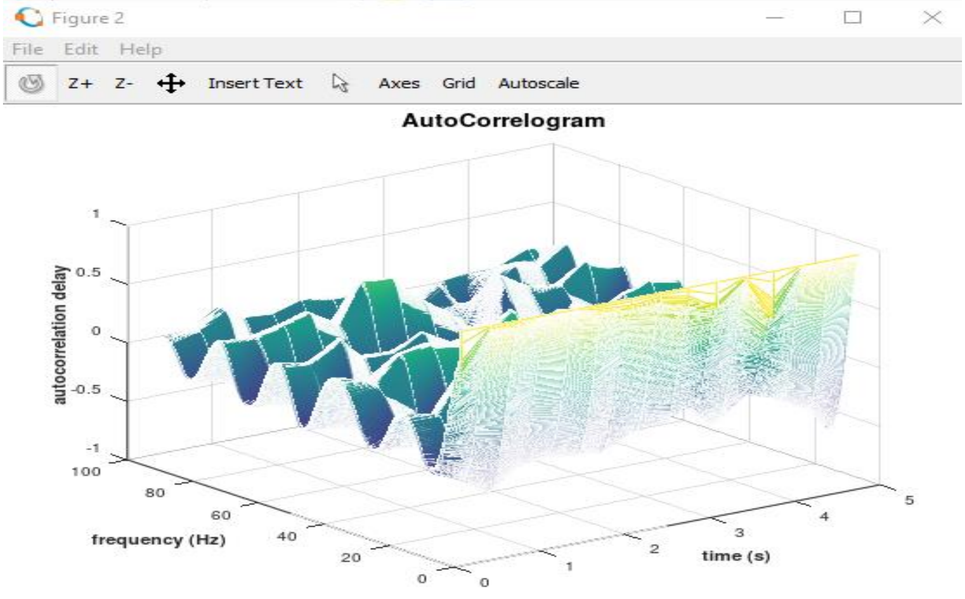


Figure 8: 3D Wave Object

Fig: Autocorelogram of "1-4211-A.ogg" of class "Fire Cackling"

Compared to the 2d spectrogram which holds only 2 dimensional features time and frequency, the autocorrelogram holds 3 dimensional features with temporal information time, frequency and autocorrelation delay. The amplitude is measured by colour intensity in both forms of visualization.

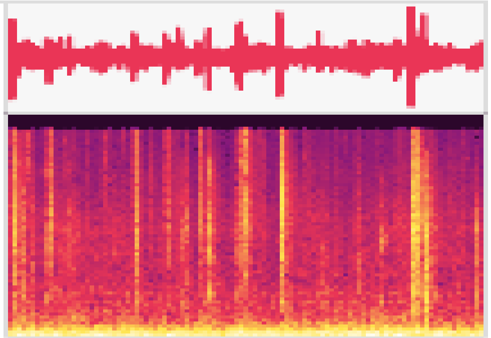


Figure 9: Spectrogram of "1-4211-A.ogg" of class "Fire Cackling"

## 5.3 CROSS FOLD VALIDATION

Primary method for estimating a tuning parameter λ (such as subset size)

- Divide the data into K roughly equal parts

| 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|
| Validation | Train | Train | Train | Train |

For each k = 1, 2 ,..., K, fit the model with parameter λ to the other K − 1 parts, giving $\hat{\beta}^{-k}(\lambda)$ and compute its error in predicting the kth part:

$$E_k(\lambda) = \sum_{i \in kth\ part} (y_i - x_i \hat{\beta}^{-k}(\lambda))^2$$

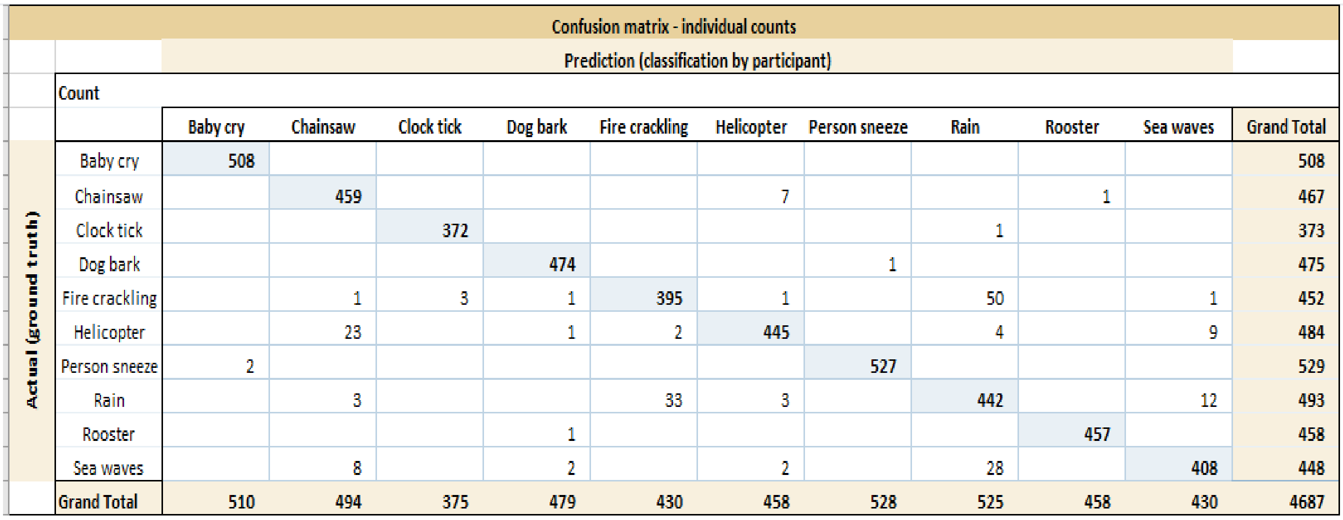
This gives the cross-validation error.

$$\text{CV}\ (\lambda) = \frac{1}{K} \sum_{k=1}^{K} E_k(\lambda)$$

- Do this for many values of λ and choose the value of λ that makes CV (λ) smallest.

Typically we use K = 5 or 10.

## 5.4 COMPARISON WITH HUMAN CLASSIFICATION:

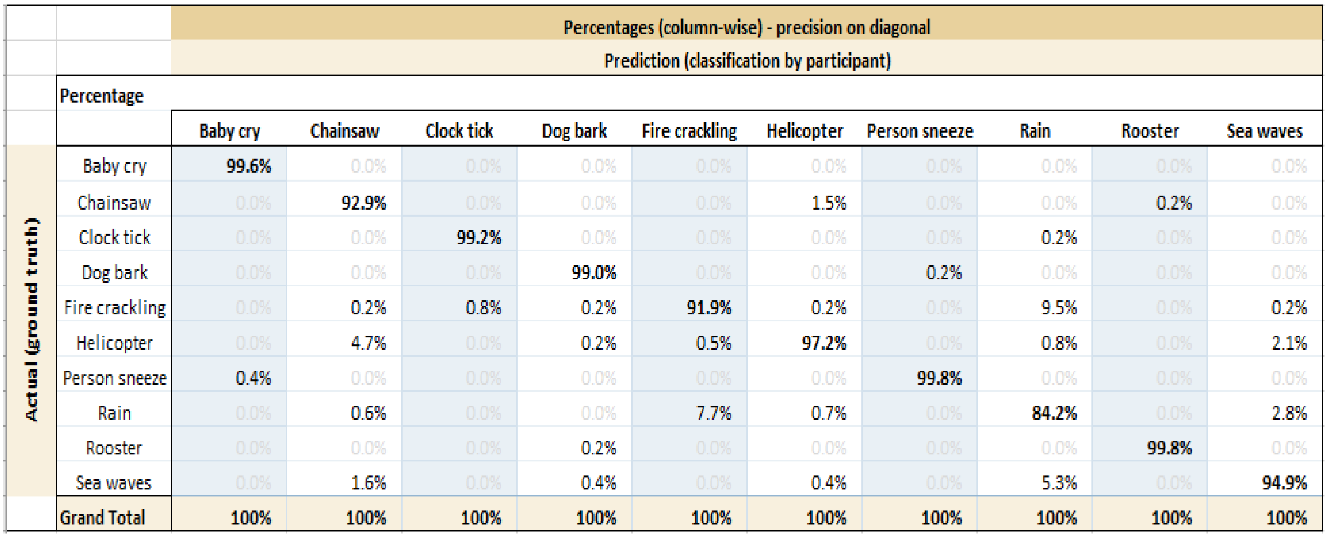


Confusion matrix - individual counts

Prediction (classification by participant)

| | Count | Baby cry | Chainsaw | Clock tick | Dog bark | Fire crackling | Helicopter | Person sneeze | Rain | Rooster | Sea waves | Grand Total |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Actual (ground truth) | Baby cry | 508 | | | | | | | | | | 508 |
| | Chainsaw | | 459 | | | | 7 | | | 1 | | 467 |
| | Clock tick | | | 372 | | | | | 1 | | | 373 |
| | Dog bark | | | | 474 | | | 1 | | | | 475 |
| | Fire crackling | | 1 | 3 | 1 | 395 | 1 | | 50 | | 1 | 452 |
| | Helicopter | | 23 | | 1 | 2 | 445 | | 4 | | 9 | 484 |
| | Person sneeze | 2 | | | | | | 527 | | | | 529 |
| | Rain | | 3 | | | 33 | 3 | | 442 | | 12 | 493 |
| | Rooster | | | | 1 | | | | | 457 | | 458 |
| | Sea waves | | 8 | | 2 | | 2 | | 28 | | 408 | 448 |
| | Grand Total | 510 | 494 | 375 | 479 | 430 | 458 | 528 | 525 | 458 | 430 | 4687 |

Figure 10: Confusion Matrix



Percentages (column-wise) - precision on diagonal
Prediction (classification by participant)

| Actual (ground truth) | Percentage | Baby cry | Chainsaw | Clock tick | Dog bark | Fire crackling | Helicopter | Person sneeze | Rain | Rooster | Sea waves |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | Baby cry | **99.6%** | 0.0% | 0.0% | 0.0% | 0.0% | 0.0% | 0.0% | 0.0% | 0.0% | 0.0% |
| | Chainsaw | 0.0% | **92.9%** | 0.0% | 0.0% | 0.0% | 1.5% | 0.0% | 0.0% | 0.2% | 0.0% |
| | Clock tick | 0.0% | 0.0% | **99.2%** | 0.0% | 0.0% | 0.0% | 0.0% | 0.2% | 0.0% | 0.0% |
| | Dog bark | 0.0% | 0.0% | 0.0% | **99.0%** | 0.0% | 0.0% | 0.2% | 0.0% | 0.0% | 0.0% |
| | Fire crackling | 0.0% | 0.2% | 0.8% | 0.2% | **91.9%** | 0.2% | 0.0% | 9.5% | 0.0% | 0.2% |
| | Helicopter | 0.0% | 4.7% | 0.0% | 0.2% | 0.5% | **97.2%** | 0.0% | 0.8% | 0.0% | 2.1% |
| | Person sneeze | 0.4% | 0.0% | 0.0% | 0.0% | 0.0% | 0.0% | **99.8%** | 0.0% | 0.0% | 0.0% |
| | Rain | 0.0% | 0.6% | 0.0% | 0.0% | 7.7% | 0.7% | 0.0% | **84.2%** | 0.0% | 2.8% |
| | Rooster | 0.0% | 0.0% | 0.0% | 0.2% | 0.0% | 0.0% | 0.0% | 0.0% | **99.8%** | 0.0% |
| | Sea waves | 0.0% | 1.6% | 0.0% | 0.4% | 0.0% | 0.4% | 0.0% | 5.3% | 0.0% | **94.9%** |
| | Grand Total | 100% | 100% | 100% | 100% | 100% | 100% | 100% | 100% | 100% | 100% |

Figure 11: Predictions for Chain Saw Class

i.e.: 92.9% of predictions when participants selected the chainsaw class, were actually for chainsaw recordings. 4.7% of chainsaw choices were actually for helicopter clips.

## VI. CONCLUSION

This paper came up with providing better accuracy when reconstructing the encoded audio files. This system used autoencoder to learn a representation (encoding) for a set of data, typically for dimensionality reduction and used LSTM networks for classification and processing and made predictions based on time series data. And also, compressed an audio file using new auto correlogram technique which provides a three dimensional view of audio wave. The resultant audio wave will have better accuracy on comparison with two dimensional view provided by existing auto correlogram technique. In existing system, the accuracy rate is 60% to 65% but when on using this system, the accuracy rate is 75% to 80%, i.e. the accuracy rate has been increased by almost 20%. In future, the accuracy rate can be further improved by using some of the finest techniques involved in RNN autoencoder and by using extremely large dataset which acts as datastore.

## REFERENCES

[1]. N.Boulanger-LewandowskiY, BengioP and Vincent, ” **Modeling temporal dependencies in high dimensional sequences: Application to polyphonic music generation and transcription,**” *Proceedings Of The Twenty-Nine International Conference On Machine Learning(2012)*

[2]. Yoshua BengioNicolas Boulanger-Lewandowski and Razvan Pascanu,” **Advances in optimizing recurrent networks** ,“ *IEEE International Conference On Acoustics, Speech And Signal Processing(2013)*

[3]. George E. DahlDong YuLi DengAlex Acero , **“Context-Dependent Pre-Trained Deep Neural Networks for Large-Vocabulary Speech Recognition,”** *IEEE Transactions On Audio, Speech, And Language Processing(2012)*

[4]. K. J. Piczak , **“Environmental sound classification with convolutional neural networks,”** *IEEE 25th International Workshop on Machine Learning for Signal Processing (MLSP)* (2015)

[5]. K. J. Piczak, **“ESC: Dataset for environmental sound classification,**” *In Proceedings of the 23rd ACM international conference on Multimedia (2015)*

[6]. H. Lee, P. Pham, Y. Largman, and A. Y. Ng, **“Unsupervised feature learning for audio classification using convolutional deep belief networks,”** *in Advances in Neural Information Processing Systems, Y. Bengio, D. Schuurmans, J. D. Lafferty, C. K. I. Williams, and A. Culotta, Eds. Curran Associates, Inc., 2009, pp. 1096–1104.*

[7]. Y.-A. Chung, C.-C. Wu, C.-H. Shen, and H.-Y. Lee, “**Unsupervised learning of audio segment representations using sequence-to-sequence recurrent neural networks,”** *in INTERSPEECH. ISCA, 2016, pp. 765–769*