# WreckingNet
# Neural approach for the classification of audio signals in construction sites

**Alessandro Maccagno**
Department of Computer, Control and Management Engineering
Sapienza University of Rome
maccagno.1653200@studenti.uniroma1.it

**Andrea Mastropietro**
Department of Computer, Control and Management Engineering
Sapienza University of Rome
mastropietro.1652886@studenti.uniroma1.it

**Umberto Mazziotta**
Department of Computer, Control and Management Engineering
Sapienza University of Rome
mazziotta.1647818@studenti.uniroma1.it

## Abstract

The aim of the project presented in this report is to create an application to recognize vehicles and tools used in construction sites, and classify them in terms of type and brand. This task will be tackled with a neural approach, that we friendly called WreckingNet, which will use different audio-based information as input to two neural networks, and then combine the respective results to improve classification certainty. One network will work on raw audio data and the other on the spectrogram of the audio source. Our architecture is based on the one from Li et al.(1), who developed a two-network architecture for environmental sound classification. Our study is different in the sense that we are focused on a very specific dataset (our work can be seen as a particular case of environmental sound recognition) and mostly, we are working on real data and not on datasets built on purpose. We will demonstrate that their architecture can be adapted with good results to a very specific domain as the one of construction sites, leading in the future to the possibility of monitoring the proper usage of machines by audio signals, for both production and safety purposes. The code of the project is available on GitHub.[1]

## 1 Dataset

The dataset we worked on is composed of real audio tracks recorded in construction sites in Utah by Professor Yongcheol Lee[2] from Louisiana State University. Unlike artificially built datasets, when working with real data different problems arise, such as noise and possibly low quality samples. Due to those complications, we focused our work on the classification of a reduced number of classes, that are *Backhoe JD50D Compact*, *Compactor Ingersoll Rand*, *Concrete Mixer*, *Excavator Cat 320E*,

[1] https://github.com/AndMastro/WreckingNet.
[2] https://www.lsu.edu/eng/cm/people/faculty/lee.php

*Excavator Hitachi 50U*, all of which having approximately 15 minutes of audio. Classes which did not have enough usable audio (too short, excessive noise, low quality of the audio) were discarded.

## 2 Data Preprocessing

In order to feed the network with enough and proper data, each audio file for each class is segmented into fixed length subsamples (the choice of the best sample size is described in the experiment section). As first step we split the original audio files into two parts, training samples (70% of the original length) and test samples (30% of the original length); this is done to avoid testing the network on data used previously to train the network, as this would cause the network to overfit and give misleading results.
Then in order to augment the dataset the files are split into smaller segments of 30ms, each of which overlaps the subsequent one by 15ms. We then compute the RMS of every signal of these smaller segments, and drop the ones with too small a power w.r.t the average RMS of the different segments, in order to remove the segments which contain mostly silence.
After that, the datasets are balanced by taking N samples for each class, where N is the number of elements contained in the class with the least amount of samples. This way, we avoided the problem of having certain classes with an abnormal number of usable audio segments being potentially either overrepresented or underrepresented and negatively impacting the training of the model, especially due to the presence of multiple models of the same vehicle.
The raw audio signal which will be fed to the first network is extracted from the files using the Python library `librosa`[3] and, using the same library, we generated the log-scaled mel spectrogram(2) of the signal that will be the input to the second network.

### 2.1 Spectrogram extraction

The technique used to extract the spectrogram from the sample is the same used by Piczak(3). The samples were re-sampled to 2*22050Hz (to pick up all of the frequencies audible by the naked human ear), then we used a window of size 1024 with hop-size of 512 and 60 mel bands. A mel band represents an interval of frequencies which are perceived to have the same pitch by human listeners. They have been found to be perofming in speech recognition.
With this parameters, and the chosen length of 30ms for the samples (see next sections), we obtain a small spectrogram of 60 rows (bands) and 2 columns (frames). Then, using again `librosa`, we compute the derivative of the spectrogram and we overlap the two matrices, obtaining a dual channel input which is fed into the network.

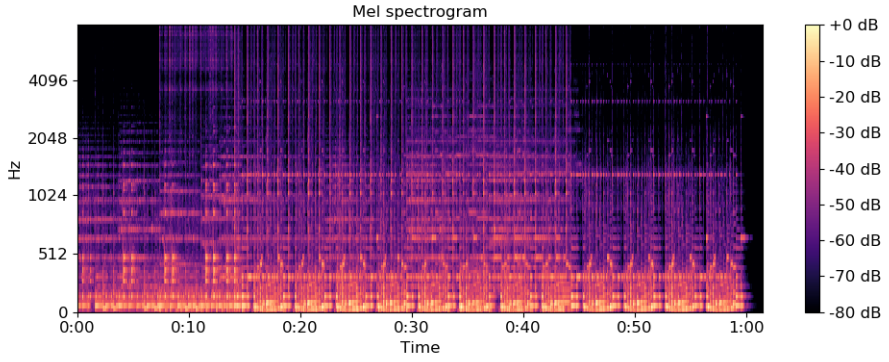


Figure 1: Example of log-mel spectrogram extracted from an audio source.

[3]https://librosa.github.io/

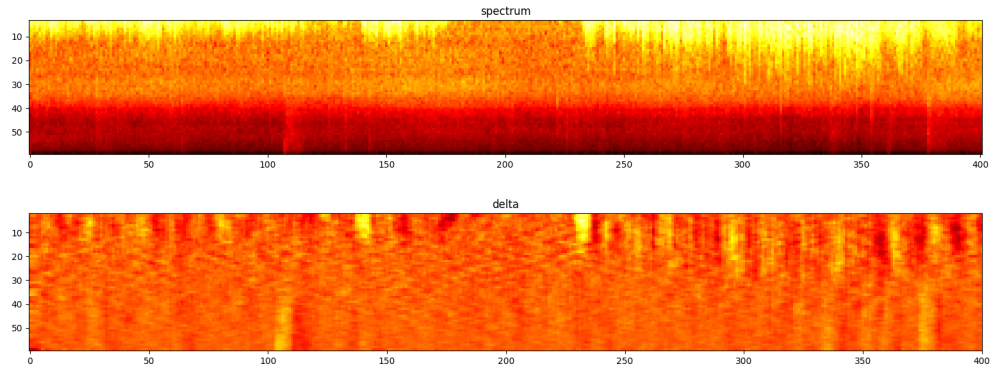


Figure 2: Our case: log-mel spectrogram extracted from a fragment along with its derivative. On the abscissae we find the time buckets, each of which representing a sample about 23ms long, while on the ordinates the log-mel bands. Since our fragments are 30ms long, the spectrogram we extract will contain 2 buckets.

## 3 Convolutional Neural Netwoks

Convolutional Neural Networks are a particular type of neural networks, which use the *convolution* operation in one or more layers for the learning process. These networks are inspired by the *primal visual system*, and are therefore extensively used with image and video inputs.
A CNN is composed by three main layers:

- **Convolutional Layer**: The convolutional layer is the one tasked with applying the convolution operation on the input. This is done by passing a filter (or *kernel*) over the matricial input, computing the convolution value, and using said result as the value of one cell of the output matrix (called *feature map*); the filter is then shifted by a predefined *stride* along its dimensions, until the entire input is covered, at which point the network repeats the operation with the next kernel.
  These filters are generally small matrices, which are trained during the training process to be able to recognize the features required by the network to solve its task.
- **Detector layer**: In the detector layer, the output of the convolution is passed through a non-linear function, usually a ReLU function, to make the learning actually possible, as otherwise a CNN would simply be a sequence of linear operations (the convolution).
- **Pooling layer**: The pooling layer is meant to reduce the dimensionality of data by combining the output of neuron clusters at one layer into one single neuron in the subsequent layer. Commonly used pooling techniques are the *Max Pooling* and the *Average Pooling*; the former outputs the maximum of the values in its receptive field, while the latter computes the average.

The last layer of the network is fully connected (a layer whose units are connected to every single unit from the previous one) that outputs the probabilty of the input to belong to each of the classes.

CNNs can improve a machine learning system since they allow:

- **Sparse Interactions**: In traditional neural networks, layers use matrix multiplication by a matrix of parameters with a separate parameter describing the interaction between each input unit and each output units and thus every output unit interacts with every input unit. In CNNs, by using kernels smaller than the input size, we let one input unit interact with few output unit, depending on the width of the kernel. This both lets us detect small features of an image such as edges and, at the same time, reduces the number of parameters to be stored while improving the statistical efficiency of the model. The output is thus computed in a faster way.
- **Parameter Sharing**: In traditional neural networks, each element of the weight matrix is used exactly once when computing the output of a layer, while in CNNs each member of

the kernel is used at every position of the input. Thus, the network does not need to learn a separate set of parameters for every location but just a single.

- **Equivariant Representations**: The particular form of parameter sharing allows the convolutional layer to be equivariant to traslation, meaning that if the input changes, the output changes accordingly. In particular, let $I(x, y)$ be a function applied to the input image and let $I'(x, y) = I(x - k, y)$ be the transformation applied to such image function. If we apply this transformation to $I$, then apply convolution, the result will be the same as if we applied convolution to $I'$, then applied the transformation to the output.

Moreover, convolutional neural networks provide a means for working with input of variable size. This is usually accomplished thanks to the pooling layers, by varying the size of an offset between pooling regions so that the classification layer always receives the same number of elements in input regardless the original input size.

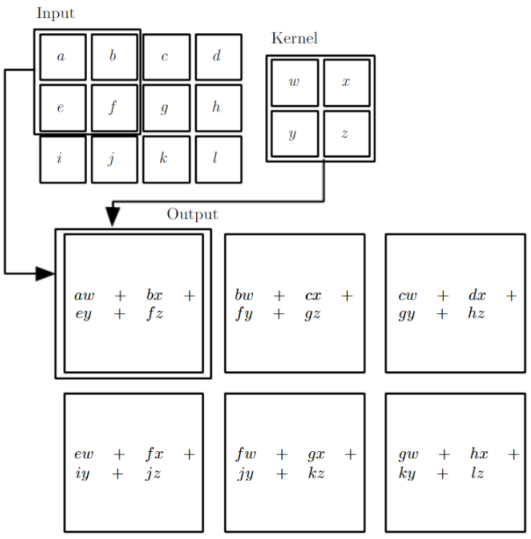


Figure 3: A schematization of the filter application process on a 2D input.(4).

### 3.1 Motivation

The reasons why we used CNNs as approach is due to the intrinsic nature of audio signals. As previously said, CNNs are extensively used with images. Thus, since the spectrum of the audio is an actual picture of the signal, it is straightforward to see why CNNs are a good idea for such kind of input. Moreover, CNNs can also be employed on raw audio data, working very well, due to their ability to capture time and frequency features.
The use of log-mel features was proposed by Piczak, as mentioned before, and the approach of extracting audio features from raw wave signals was employed by Tokozume et al.(5). The need to combine the two approaches lies in some limitation they have if applied independently. Firstly, log-mel features were originally designed for ASR (Automated Speech Recognition) rather then ESR (Event Sound Recognition) and thus may fail in capturing some information of sound events that may be critical. Secondly, the end-to-end approach from raw data performs always worse than the log-mel counterpart.
Instead, by combining the two approaches, classification performance improves, as we will show.

## 4 Architecture

As aforementioned, WreckingNet architecture is composed as follows:

1. **RawNet**: Convolutional neural network which uses the waveform of the audio segment as input.

2. **SpectroNet**: Convolutional neural network which takes as input the concatenation of the spectrum of each audio segment along with its time derivative.
3. **DSE Module**: The last component of the application, it receives the label probabilities output from the two previous networks and combines them, returning the final probability distribution used to classify a sample, by applying the Dempster–Shafer Evidence theory.

### 4.1 RawNet

This CNN works directly on the raw waveform of the audio data, and it is composed of the following layers:

1. Input layer: size of (1,662).
2. Convolutional layer: 40 filters, kernel size of (1, 8), strides (1,1), ReLu activation function.
3. Convolutional layer: 40 filters, kernel size of (1, 8), strides (1,1), ReLu activation function.
4. Max pooling: pooling size of (1, 128), strides (1,4).
5. Convolutional layer: 24 filters, kernel size of (6, 6), strides (1,1), ReLu activation function.
6. Convolutional layer: 24 filters, kernel size of (6, 6), strides (1,1), ReLu activation function.
7. Convolutional layer: 48 filters, kernel size of (5, 5), strides (2,2), ReLu activation function.
8. Convolutional layer: 48 filters, kernel size of (5, 5), strides (2,2), ReLu activation function.
9. Convolutional layer: 64 filters, kernel size of (4, 4), strides (2,2), ReLu activation function.
10. Dense layer: 200 units, ReLU activation function.
11. Dropout: dropout rate of 0.3.
12. Output layer: dense layer with 5 units (one for each class) with softmax activation function.

The network employs an Adam Optimizer(6) with a learning rate of 0.0005.

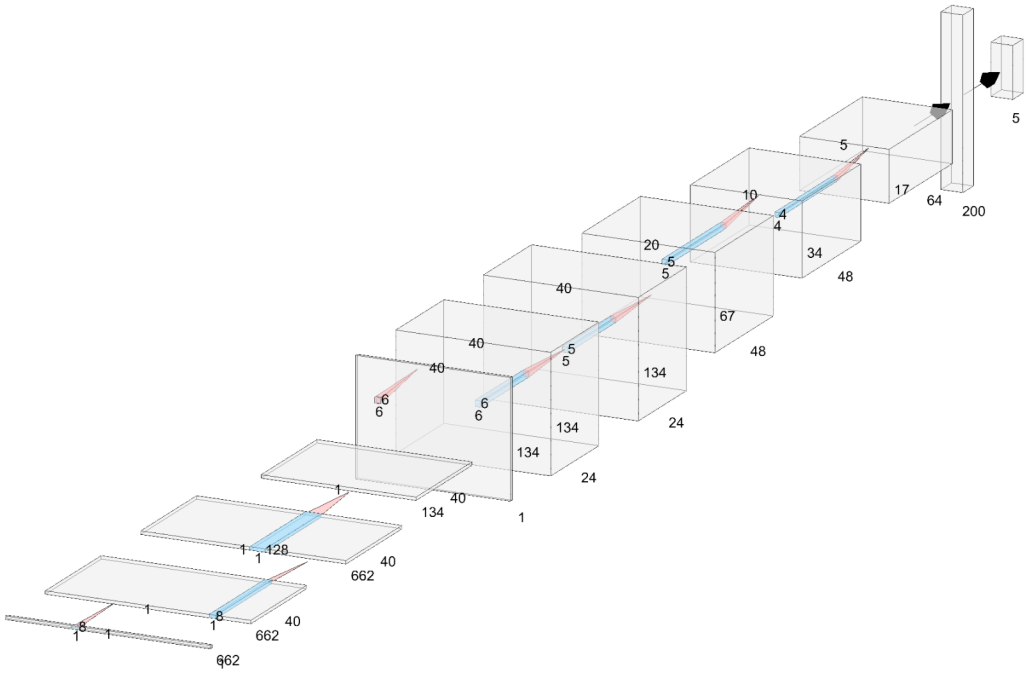


Figure 4: RawNet architecture.

### 4.2 SpectroNet

This CNN is fed with the spectre of a sample concatenated with its time derivative:

1. Input layer: size of (60,2,2).
2. Convolutional layer: 24 filters, kernel size of (6, 2), strides (1,1), ReLu activation function.
3. Convolutional layer: 24 filters, kernel size of (6, 2), strides (1,1), ReLu activation function.
4. Convolutional layer: 48 filters, kernel size of (5, 1), strides (2,2), ReLu activation function.
5. Convolutional layer: 48 filters, kernel size of (5, 1), strides (2,2), ReLu activation function.
6. Convolutional layer: 64 filters, kernel size of (4, 1), strides (2,2), ReLu activation function.
7. Dense layer: 200 units, ReLu activation function.
8. Dropout: dropout rate of 0.3.
9. Output layer: dense layer with 5 units (one for each class) with softmax activation function.

SpectroNet uses an Adam Optimizer as well, with the same learning rate of 0.0005.

Regarding the learning rate, we performed several runs trying different values (such as 0.001, 0.0001, 0.00001 and more) but we ended up having the best results with 0.0005.

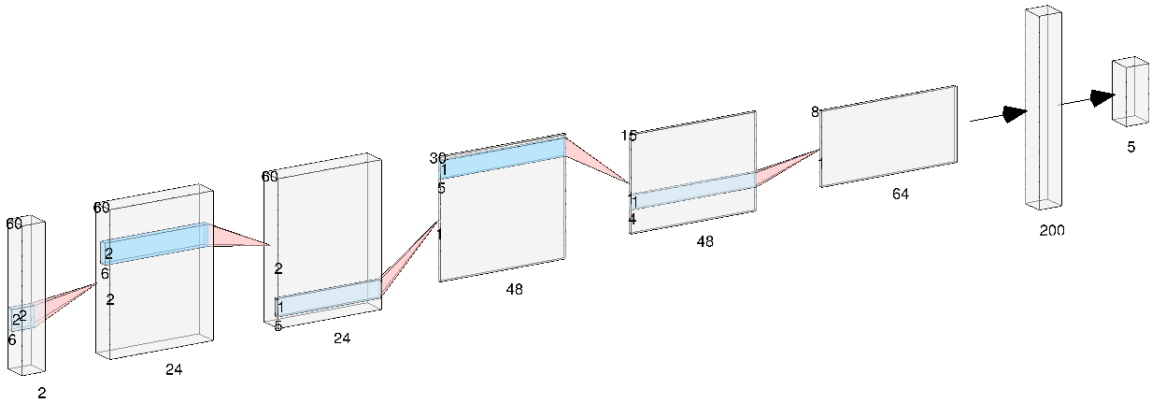


Figure 5: SpectroNet architecture.

### 4.3 DSE Module

The DSE module is the final component of our application, and it is tasked with combining the resulting class probability distributions outputted by the two different networks. This is achieved through the Dempster-Shafer theory, a reasoning framework commonly used to combine beliefs produced by different sources.
Given the set of all the possible (mutually exclusive) states of the problem

$$S = \{C_0, C_1, C_2, ...\} \tag{1}$$

and its power set $2^S$, made of all of the possible subsets of the universe set, we define as *mass* of an item the function

$$m : 2^S \rightarrow [0, 1] \tag{2}$$

which represents the weight given to the particular state, which will be used to calculate its *belief* in a later step. This function has two important properties:

$$m(\emptyset) = 0 \tag{3}$$

$$\sum_{C \in 2^S} m(C) = 1 \tag{4}$$

All of the functions with same domain and codomain as the mass function which follow these two properties are called *basic belief assignments* (BBAs).
In our application, the universe set S contains the classes chosen from the dataset, which are mutually exclusive per problem construction, since every audio records a single vehicle, while the masses of each class are represented by the output of the softmax of the two networks (with $m_1$ being the RawNet and $m_2$ being the SpectroNet). Since the two softmax outputs are probability distributions, we are feeding to the DSE module elements which follow the BBA properties, meaning that, since our problem respects the properties of the theory, we can apply the following DS combination rule without risk of error:

$$m(C) = \begin{cases} 0 & \text{if C = Ø} \\ \frac{1}{K} \sum_{C_1 \cap C_2 = C} m_1(C_1) m_2(C_2) & \text{otherwise} \end{cases} \tag{5}$$

Where $K$ is calculated as:

$$K = \sum_{C_1 \cap C_2 = \emptyset} m(C_1) m(C_2) \tag{6}$$

The first formula defines the combined mass of a single item as the sum over the product between the two different mass functions (product which will be henceforth be referred to as 'combined mass'), repeated for each couple of items whose intersection is our starting item itself, scaled by $K$. This can be intuitively interpreted as the sum of the masses for every possible way in which the state C may appear in our problem. The factor $K$ instead represents the amount of conflicts present in the problem, where a conflict is considered as a couple of items with no sub-elements in common; the higher the combined mass, the more the two function give different results, and therefore the probabilities should be 'trusted' less.
In the case of this application, the masses, being softmax outputs, are encoded as n-elements-long lists (with n being the number of classes), where an element in position $i$ corresponds to the network's confidence that the audio segment's class is the $i$-th. Since classes are mutually exclusive, the only combinations which give a non-zero combined mass are the ones comprising equal states (i.e. equal classes), reducing the problem to an element-wise product between the two softmax outputs, scaled by the $K$ factor obtained with formula (6), and then normalized to obtain an actual probability distribution.
The result of this operation was immediately visible upon testing: by combining the network outputs we increased the overall confidence of the system and corrected possible errors made by the networks, leading to an increase in accuracy of an average 4% for each class.

## 5 Experiments

The experiments were run on 4 different machines:

- Laptop Intel core i7-7700HQ, 16GB RAM DDR4, NVIDIA GeForce GTX1060 6GB GDDR5.
- Laptop Intel core i7-7500U, 16GB RAM DDR4, NVIDIA GeFotrce GTX950M 2GB GDDR5.
- Laptop Intel core i7-4410HQ, 16GB RAM DDR3, NVIDIA GeForce GTX860M 4GB GDDR5.
- Desktop Intel core i7-3770, 16 GB RAM DDR3, NVIDIA GeForce GTX 970 4GB GDDR5.

The use of different machines proved to be crucial for the execution of the cross validation, since training the network multiple times on a single machine would have taken too long.

### 5.1 Experiment Setup

A sizeable amount of time in the project was spent into finding the proper length for the audio segments. This is of crucial importance since if the length is not adequate the network will not find any useful features to learn.
Since the generation of the dataset by both splitting the audio files and generating the spectrogram takes a lot of time, we decided firstly to work only with the RawNet in order to determine the proper segment size. We generated different dataset variants by splitting the audio using different lengths and trained different models. The testing results are show in fig 6.

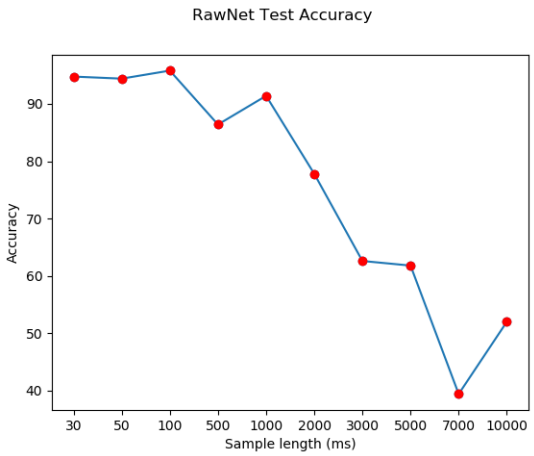


Figure 6: Accuracy according different sample sizes on RawNet.

As we can see, with smaller samples sizes we obtain better results, while we notice a drop as the size increases. It is also interesting to observe that with very large sample sizes the accuracy tends to improve a little again; it could be due to the fact that with long signals the network tends to learn just the mean of the signal values, being able to recognize more samples but not leading to anything interesting in the field of deep learning, since the mean of a signal can be computed in simpler ways. Finally, we ended up choosing 30ms as sample size, since it led not only to having a high accuracy but also a larger number of samples.
In order to properly test the network we performed a k-fold cross validation, with $k = 5$. The results of the classification are shown in the next section.

### 5.2 Classification results

Table 5.2 shows the classification accuracy of the two networks alone and of the combination of the two network using DSE (WreckingNet).

| Network | Accuracy | Precision | Recall | F1 |
|---|---|---|---|---|
| RawNet | 93.59 | 93.68 | 93.55 | 93.61 |
| SpectroNet | 97.08 | 97.34 | **97.30** | 97.32 |
| WreckingNet (DSE) | **98.27** | **97.84** | 97.10 | **97.46** |

Table 1: K-Fold Cross Validation Classification Result

As we can notice, by applying the DSE, we obtain results that are higher than the highest of the results of the two networks alone, except for the recall that is slightly higher with the SpectroNet, even though really close to the value obtained by the combination of the networks. Figures 7 and 8 show the accuracy of the two networks and the confusion matrix of WreckingNet.

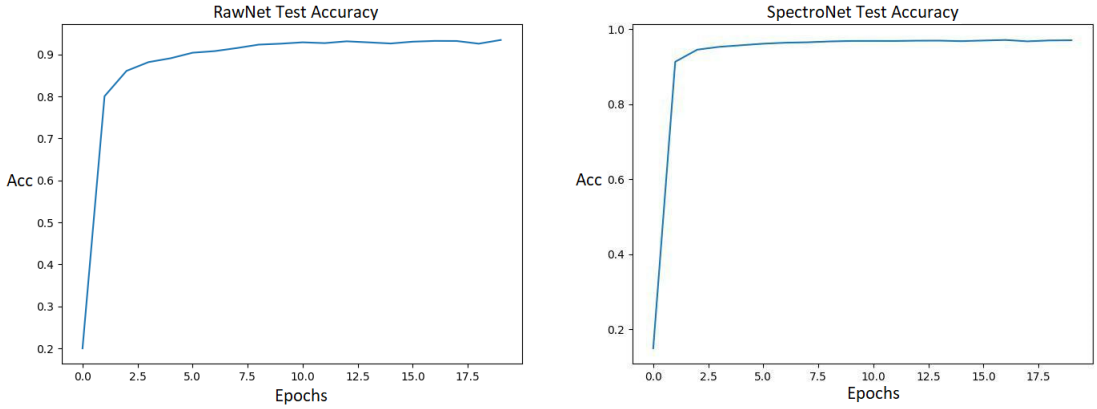


Figure 7: Accuracy on test set.

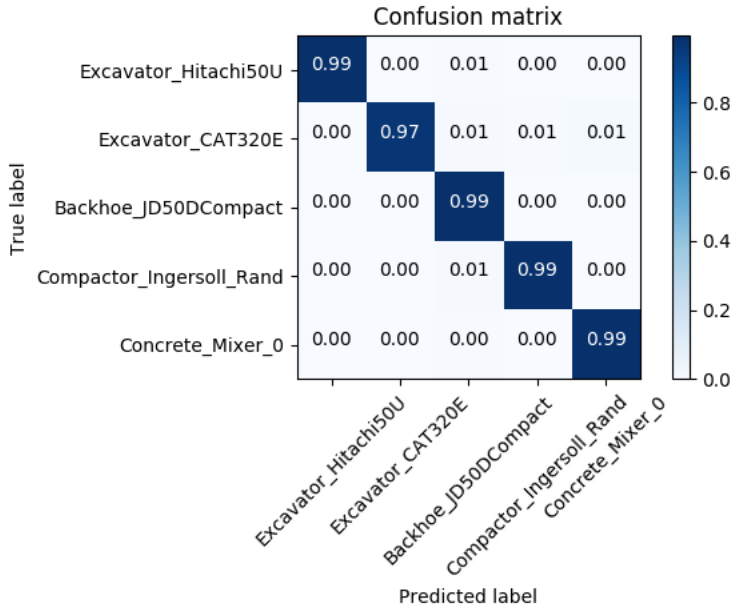


Figure 8: Confusion matrix for WreckingNet.

### 5.3 Prediction

In order to predict a new sample in input, such audio file is split into segments as described above; every fragment will be classified as belonging to one of the classes and the audio track will be labelled according to the majority of the labels among all the fragments; in this way we can also see what is the probability for the input track to belong to each of the classes.

## 6 Conclusions and Future Work

In this paper we demonstrated that it is possible to apply a neural approach already tested in environmental sound classification to a more specific domain, that is the one of construction sites, with impressively high results.
Up to now, the network was tested on 5 classes; the idea is to try to increase the number of classes to include more tools and vehicles employed in building sites in order to lead in the future to a system able to identify a large variety of classes for both monitoring the usage of machines and safety purposes in case of sudden malfunction. Our idea is to try to improve our work as part of the *Honors Program*, under the supervision of Professor Scarpiniti[4] and collaborating with Professor Yongcheol Lee.

## References


[1] S. Li, Y. Yao, J. Hu, G. Liu, X. Yao, and J. Hu, "An ensemble stacked convolutional neural network model for environmental event sound recognition," *Applied Sciences*, vol. 8, no. 7, 2018.

[2] E. B. Stevens, Stanley Smith; Volkmann; John Newman, "A scale for the measurement of the psychological magnitude pitch," 1937.

[3] K. J. Piczak, "Environmental sound classification with convolutional neural networks," in *2015 IEEE 25th International Workshop on Machine Learning for Signal Processing (MLSP)*, pp. 1–6, Sep. 2015.

[4] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[5] Y. Tokozume and T. Harada, "Learning environmental sounds with end-to-end convolutional neural network," in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 2721–2725, March 2017.

[6] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *ICLR*, 2015.


[4] http://michelescarpiniti.site.uniroma1.it