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

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

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

Umberto Mazziotta
Department 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, 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 [INSERT PROF NAME HERE]. Conversely to datasets artificially built, working with real data different problems arise, such as noise and low quality data. Due to this complications, we focused our work to the classification of a reduced number of classes, that are *Backhoe JD50D Compact*, *Compactor Ingersoll Rand*, *Concrete Mixer*, *Excavator Cat 320E*, *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.

[1] GitHub link: https://github.com/AndMastro/WreckingNet.

Neural Networks Project, A.Y 2018/2019, Sapienza Univerisy of Rome.

2 Data Processing

In order to feed the network with enough and proper data, each audio file for each class is segmented into fixed length samples (the choice of the best sample size is described in the experiment section), from which we remove the ones with too much or too little noise w.r.t. the average. As first step 23 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 data set 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 element contained by 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` [insert cite here] and, using the same library, we generated the log-scaled mel-spectrogram 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 [ADD REF HERE]. The samples were re-sampled to 22050Hz (to cover 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. 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.

3 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 spectre 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.

3.1 RawNet

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

1. Convolutional layer: 40 filters, kernel size of (1, 8), strides (1,1), ReLu activation function.
2. Convolutional layer: 40 filters, kernel size of (1, 8), strides (1,1), ReLu activation function.
3. Max pooling: pooling size of (1, 128), strides (1,1).
4. Convolutional layer: 24 filters, kernel size of (6, 6), strides (1,1), ReLu activation function.
5. Convolutional layer: 24 filters, kernel size of (6, 6), strides (1,1), ReLu activation function.
6. Convolutional layer: 48 filters, kernel size of (5, 5), strides (2,2), ReLu activation function.
7. Convolutional layer: 48 filters, kernel size of (5, 5), strides (2,2), ReLu activation function.

8. Convolutional layer: 64 filters, kernel size of (4, 4), strides (2,2), ReLu activation function.
9. Dense layer: 200 units, ReLu activation function.
10. Dropout: dropout rate of 0.3.
11. Output layer: dense layer with 5 units (one for each class) with softmax activation function.

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

3.2 SpectroNet

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

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

The architecture is the same of the RawNet except for the first three layers that are missing. It 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 bu having the best result with 0.0005.

3.3 DSE Module

The DSE module is the final component of our application, and it is tasked with combining the resulting class probability distributions output 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

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

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 DSA module elements which follow the BBA properties, meaning that We can therefore 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.

4 Experiments

4.1 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. [add interesting audio stuff]
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. The results are show in fig 1.

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; 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.

4.2 Classification results

Table 4.2 shows the classification accuracy of the two networks alone and of the combination of the two network using DSE.

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 higher with the SpectroNet, even though really close to the value obtained by the combination of the networks.

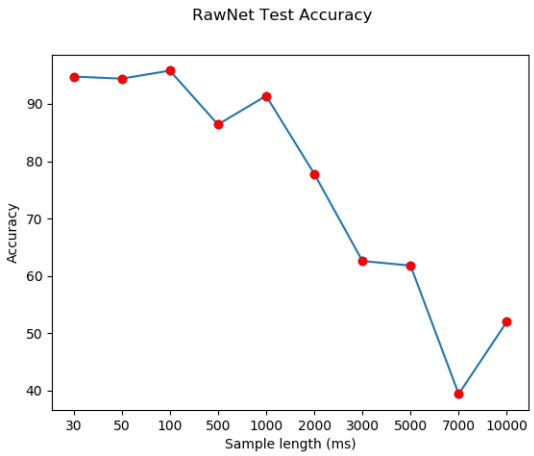

Figure 1: Accuracy according different sample sizes on RawNet.

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

Table 1: K-Fold Cross Validation Classification Result

4.3 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 label

5 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 pretty high results.
Up to now, the network was tested on 5 classes; the idea is to try to extend the classes to include more tools and vehicles employed in building sites such that to lead in the future to a system able to identify a large variety of classes for both monitoring the usage of machines and for safety purposes in case of sudden malfunctioning. Our idea is to try to improve our work as part of the *Honour Program*, followed by Professor Scarpiniti and also working with the American Professor [INSERT NAME HERE].

References

[1] Li, S.; Yao, Y.; Hu, J.; Liu, G.; Yao, X.; Hu, J. An Ensemble Stacked Convolutional Neural Network Model for Environmental Event Sound Recognition. Appl. Sci. 2018, 8, 1152.

[2] Librosa python library: https://librosa.github.io/

[3] K. J. Piczak, "Environmental sound classification with convolutional neural networks," 2015 IEEE 25th International Workshop on Machine Learning for Signal Processing (MLSP), Boston, MA, 2015, pp. 1-6. doi: 10.1109/MLSP.2015.7324337