Audio classification on a deaf machine

Bernhard Ertel
University of Innsbruck
`bernhard.ertel@student.uibk.ac.at`

Abstract

We present a novel method for transforming audio clips into melspectrograms for classification. The classification algorithm we used is a convolutional network similar to the VGG-Net. The results seem promising with accuracy up to 80 % and an average of 74.1 % using cross validation. The transformed and original dataset as well as the implementation can be found here: git.uibk.

1 Introduction

Convolutional and capsule networks show amazing results on image tasks. There are several pre-trained networks publicly available that can be used for transfer learning.

The idea presented here is to convert audios into images to then do image classification on a convolutional network. The architecture of the network that we used is similar to the VGG-Net that produces good results on classification tasks.

2 Previous work

2.1 DataSet

The dataset that is used in this project is called *UrbanSound8K*[1]. It consists of 8732 audio clips with a length of maximum 4s with the classes: *air conditioner, car horn, playing children, dog bark, drilling, engine idling, gun shot, jackhammer, siren and street music.* These audio clips are subclips from longer audios. From each audio there can be multiple sub clips in the data. The dataset is divided into 10 folds where each fold contains subclips from different audio sequences.

2.2 Classification

In [1] different algorithms have been applied to the samples from the dataset using 40 different frequencies between 0 and 22050 Hz. The results for using different length of audio is shown in figure 1.

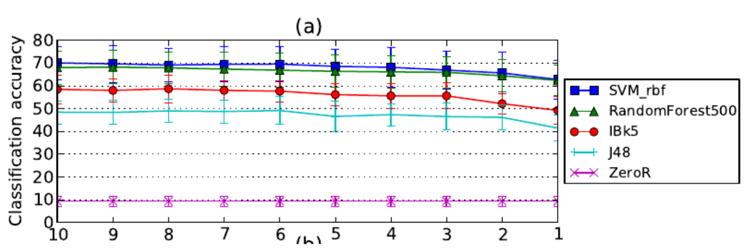

Figure 1: The classification accuracy vs Maximum slice duration in seconds for different classifier algorithms.[image from [1]]

In [2] they use CNNs and data augmentation to classify the audios with an accuracy of up to 85% on this dataset (see figure 2).

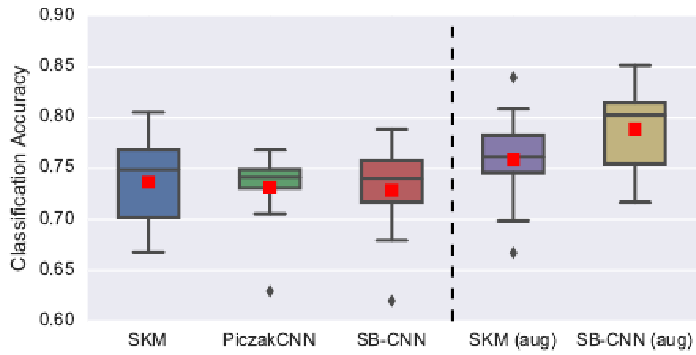

Figure 2: Classification accuracy for different architecture with (right) and without data augmentation (left). [image from [2]]

2.3 Using Melspectrogram on audio tasks

In a previous work on *Speech Emotion Recognition From 3D Log-Mel Spectrograms With Deep Learning Network* [3] they use 3 different spectrograms for analysis on audio with convolutional network architectures. The underlying spectrums are a Log-Mel spectrum and the Log-Mel spectrums' deltas and delta-deltas.

3 Melspectrogram

A spectrogram of sound is the spectrum of frequencies over time. To get the spectrum of frequencies out of the signal *short-time fourier transformation*[STFT][4] is applied. In our case the length of every window in the STFT is 512. The number of analysed frequencies is 1025.

A Melspectrogram is a spectrogram with the mel-scale on the y-axis. The conversion from Hz into Mel that is used here is[5]:

$$m = 2595 \log_{10}\left(1 + \frac{f}{700}\right). \tag{1}$$

Thus in the mel-scale the higher frequencies are closer together. In our transformation from spectrogram to mel-scale spectrogram the number of frequencies decreases from 1025 to 128.

4 Methodology

4.1 Data processing

In the first step we create the mel spectrogram using the librosa-library[6]. The steps are shown in figure 3 for three audio examples.

The resulting array S has 128 rows corresponding to different frequencies. In the next step I normalize the values with *librosa.power_to_db()*. This rescales the array S in the following way:

$$S_n = 10 \log_{10}\left(\frac{S}{|max(S)|}\right) \tag{2}$$

Because the values in S_n are ranging from -80 to 0 I rescale them to fit the rgb-scale from 0 to 255. Some of the audio files are really short (less than a second). To get all images being the same size I just stack the input multiple times until the width reaches the height. Using *PIL*-library we created images that were saved as jpg-files. This compresses the dataset from 7.1 GB down to 34MB which makes it possible to use Google Colab for training the network.

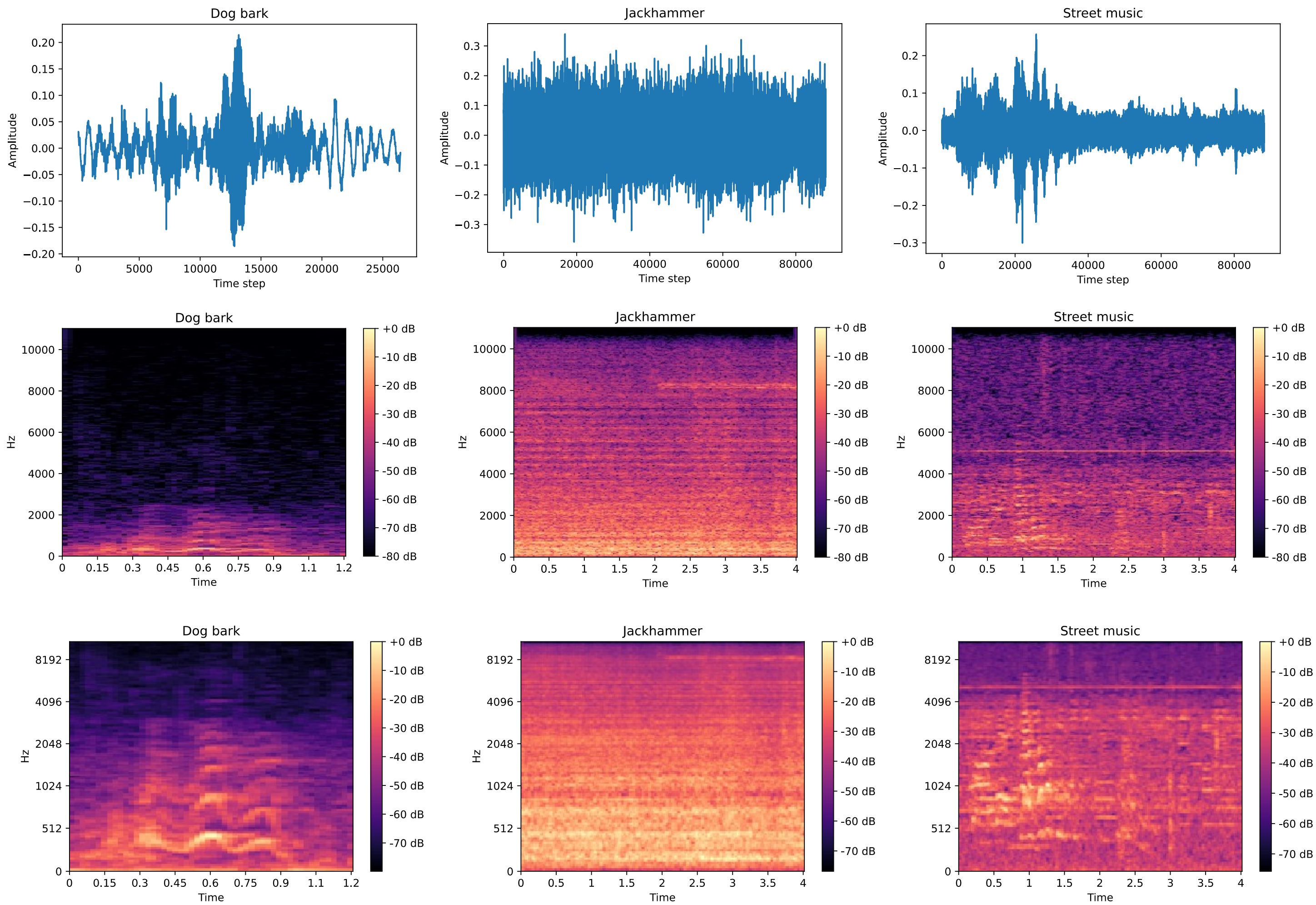

Figure 3: The audio signal for three examples in the first row, the second row shows the spectrogram and the third row the mel-scaled spectrogram. The three samples can be played by clicking: [dog bark] , [jack hammer] , [street music]

In the next step the images get cropped so that they are quadratic. After evaluating mean and standard deviation of the whole dataset I normalize the images.

4.2 Algorithm

We use a slightly modified vgg16:[7] with one instead of three input layers for the spectrogram and 10 output neurons, one for each class.

4.3 Training

We use a Stochastic Gradient Descent [SGD] optimizer with a learning rate of 0.001 and a momentum factor of 0.9. The loss criterion we used is Cross Entropy Loss as we have 10 different classes in the dataset.

5 Results

In the first trial we did a random split into test and training data and achieved accuracy of 91% which sounds great but in the end is not comparable to other work. This good result comes from the structure of the dataset as described earlier.

Using 10-fold cross validation to evaluate the performance we get the following results:

Validation on fold#	Accuracy in %
1	72
2	69
3	69
4	70
5	76
6	74
7	78
8	73
9	80
10	80
Average	**74.1**

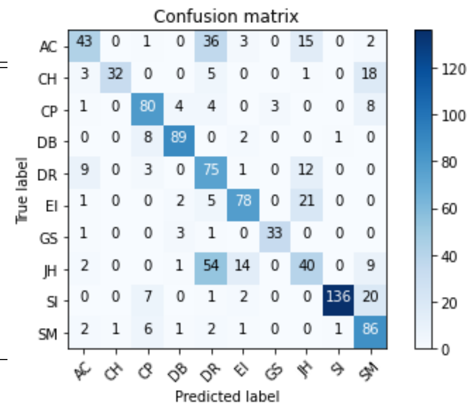

Figure 4: Accuracy on the different folds (left) and the confusion matrix of fold 4.

6 Discussion

Looking at the confusion matrix in figure 4 one can clearly see that the model has problems with some of the classes. For example it happens often that the jackhammer is classified as a drilling machine, which is not surprising since the Mel spectrograms in figure 5 for jackhammer and drilling machine are very similar. Distinguishing between the two classes can be hard even for humans listening to the sounds.

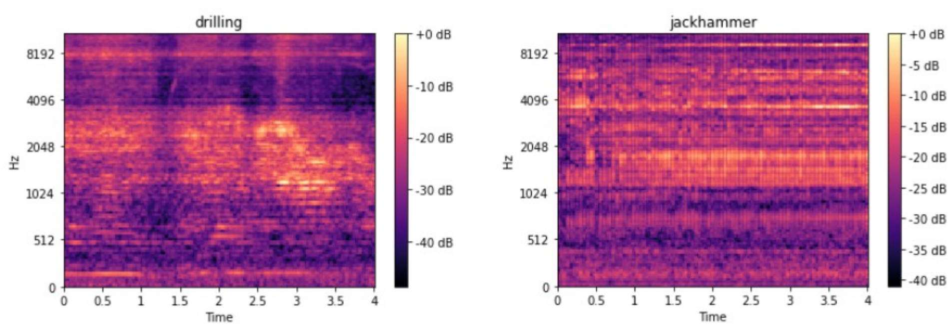

Figure 5: Mel-spectrogram of a jackhammer and a drilling machine compared to each other. audio: [drilling machine] [jack hammer] ,

Low computation power and the conditions of Google Colab (you cannot train a model for more than 12h) made it hard for to improve the results further. But anyways it is a success to achieve results that compare to the ones from the papers mentioned earlier. This shows the importance of data transformation to make it understandable for specific algorithms.

7 Future work

To improve the accuracy further it surely makes sense to use data augmentation by adding noise (as they did in [2]) to the samples and fine tune the parameters further.

Adding another channel with different frequencies or the deltas as described in [3] could have further impact and one would be able to test the performance on pretrained networks. (This was not possible as we have only one input channel)

Training other architectures like a Recurrent Neural Network or a Capsule Network to compare the results to our result could lead to some interesting insights.

References

[1] Justin Salamon; Christopher Jacoby; Juan Pablo Bello. Urban Sound Datasets. *MM '14 Proceedings of the 22nd ACM international conference on Multimedia*, (3):1041–1044, 2014.

[2] Justin Salamon and Juan Pablo Bello. Deep Convolutional Neural Networks and Data Augmentation for Environmental Sound Classification. *IEEE Signal Processing Letters*, 24(3):279–283, 2017.

[3] Hao Meng, Tianhao Yan, Fei Yuan, and Hongwei Wei. Speech Emotion Recognition from 3D Log-Mel Spectrograms with Deep Learning Network. *IEEE Access*, 7:125868–125881, 2019.

[4] Short-time fourier transform. `https://en.wikipedia.org/wiki/Short-time_Fourier_transform`, 2021. Accessed: 2021-01-19.

[5] Source code for librosa.core.convert. `https://librosa.org/doc/latest/_modules/librosa/core/convert.html#hz_to_mel`, 2021. Accessed: 2021-01-19.

[6] audio and music processing in python. `https://librosa.org`, 2021. Accessed: 2021-01-19.

[7] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*, pages 1–14, 2015.