UNIVERSIDAD TECNICA
FEDERICO SANTA MARIA

Replication Project:

Deep Convolutional Neural Networks and Data Augmentation for Environmental Sound Classification[1]

Subject: IPD423 - Procesamiento Avanzado de Señales

Professor: Matías Zañartu, Ph.D.

Due date: June 15, 2020

Student: Gabriel Rudloff Barison
Rol: 201303044-0

1 Abstract

The objective of this work is to reproduce and propose an improvement over the results presented by Justin Salomon and Juan Pablo Bello on their paper "Deep Convolutional Neural Networks and Data Augmentation for Environmental Sound Classification" [1]. Their work stems from the success of convolutional neural networks (CNN) on the image classification task, transferring it to the audio event detection scenario by first computing the log melspectrogram of the audio signals. They propose a CNN model and assert its capacity over the UrbanSound8K [2] dataset. Furthermore they propose the use of data augmentation for improving the variability and size of the dataset by applying transformations that conserve the original sound semantics. They show that the combination of their model together with data augmentation achieves state-of-the-art and outperforms a "shallow" dictionary learning approach over the same augmented dataset, as well as the same models over the original dataset. One evident limitation of their model is their performance over noise-like audio, to improve this we propose the use of the discrete wavelet transform over the original feature space in order to achieve a channel-wise separation of low and high frequencies of the spectrogram image.

2 Introduction

The authors aim to make a contribution on automatic environmental sound classification, which consist on models that can accurately discern between samples from a set of sound classes normally present in an environment, in particular they propose one for the context of an urban environment.

They propose the use of CNN for this task for two main reasons. Firstly, their capacity to characterize spectro-temporal patterns which are an important feature to be able to discern between noise-like classes. Secondly, by using low size convolutional kernels this models can detect large-scale patterns as a composition of small-scale patterns over each layer, the authors propose that this yields a more robust classification capacity even if part of the sound is masked by other sources.

Even though deep learning models are highly capable of constructing a mapping from input feature space to the class set, they are highly dependent on the size of the dataset and the inter-class variability. The task at hand doesn't have as much popularity as compared with the image classification, consequently there is a limited number of available datasets and with data sizes considerably lower. One way of tackling this issue is the use of data augmentation, which is a very popular technique in the image classification realm. It consist on yielding additional data by the deformation of original data through semantic-preserving deformations. Through this the network should become invariant to the applied deformations, and in doing so achieve a more general representation of the data.

The rest of the document is structured as follows. On section 3 the method of the original paper is explained and how we intent to reproduce it and the differences of our implementation, as well as the improvement proposition. On the section 4 we show the original results from Salomon & Bello and compare them with our results on each scenario. Finally on 5 we discuss

possible explanations for the discrepancies on the replicated results as well as the main insights from the results, we also discuss the value of our improvement approach. There is an appendix on section 6 where we explain some general DSP concepts and other miscellaneous subsections.

3 Method

3.1 Input Feature Space.

The datasets consists of 8000 samples with varying length up to 4 seconds, the chosen input feature space is the log-scaled mel-spectrogram. See 6.1 for a detailed description of how this are constructed.

The feature space representation is constructed with 128 bands covering the human audible frequency range $[0 - 22050]$Hz, using a window size of 1024 samples ($\approx 23[ms]$ at $44.1kHz$) and a hop of the same size. Using these settings, a 3 seconds signal yields 128 frames, making the input feature space $X \in \mathbb{R}^{frames \times bands}$, with $frames = 128$ and $bands = 128$.

The authors tackle the varying length issue by extending the samples with lengths shorter than 3 seconds by repetition[1], then during training a random 128 frames window is taken on every epoch and in testing the output is the average over every possible 128 frames window.

On the original work *essentia* is used to extract the log-scaled mel-spectrogram, in contrast we use *librosa* for this.

3.2 Deep Convolutional Neural Network

3.2.1 Original Architecture

The architecture described by the authors is composed of 3 convolutional layers, followed by two fully connected layers. A visualization is displayed in the figure 1, the output shapes of every layer can be observed from it. The details of every layer, such as the number of filters, their activation, the regularization methods are described below.

Input: (128, 128, 1) TF-patch taken from the log-scaled mel-spectrogram

ℓ_1: Conv2D: Convolutional layer with 24 filters and a (5,5)-sized kernel, this yields a weights matrix of size (24,5,5,1)
MaxPool2D: Max Pooling with shape (4,2) and stride of the same size.
Activation: Rectified linear unit activation (Relu).

ℓ_2: Conv2D: Convolutional layer with 48 filters and a (5,5)-sized kernel, this yields a weights matrix of size (48,5,5,24)
MaxPool2D: Max Pooling with shape (4,2) and stride of the same size.
Activation: Relu.

[1] J. Salomon was kind enough to answer this for me as it was not said in their paper

ℓ_3: Conv2D: Convolutional layer with 48 filters and a (5,5)-sized kernel, this yields a weights matrix of size (48,5,5,48)
Activation: Relu.
Regularization: Dropout with 0.5 probability.

ℓ_4: Flatten: The multidimensional array is flattened to only have one dimension in order to be admissible by the next layer.
Dense: Fully connected layer with 64 nodes. Weights matrix has shape (2400,64).
Activation: Relu.
Regularization: ℓ_2 regularization with $\alpha = 0.01$ and dropout with 0.5 probability.

ℓ_5: Dense: Fully connected layer with 10 nodes. Weights matrix has shape (64,10).
Activation: Softmax.
Regularizatión: ℓ_2 regularization with $\alpha = 0.01$

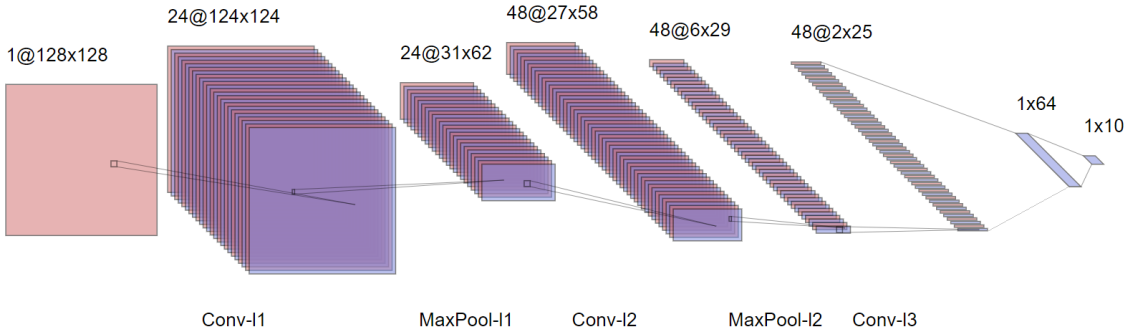

Figure 1: Salomon et al. CNN

3.2.2 Training

For the training the loss is chosen to be the categorical cross entropy, it is minimized through mini-batch stochastic gradient descend with $batch\_size = 100$ and $learning\_rate = 0.01$.

As it is mentioned in 3.1, the authors train over random 3 second excerpts from longer samples on every epoch, and during testing and validation they output the average over all possible 3 second excerpts. This adds a significant overhead on the data streaming of the training process and because of this it is chosen to be simplified to just taking the first 3 seconds of every window. Nonetheless we try to assert the impact of this by training the model on the non-augmented dataset with and without this "data streaming overhead".

The authors train the model over 50 epochs, on each they take random mini-batches until 1/8 of all the data is exhausted (referring to all possible excerpts from all the samples and their augmentations), then they chose the epoch that yields the best parameters according to the accuracy on the validation fold. In contrast we use early-stopping, which consists on stopping once the validation loss stops decreasing, and then the parameters of the epoch with the minimum validation loss are chosen. In particular we use early-stopping with $patience = 15$, this

means that once a minimum value of validation loss is obtained it will wait 15 steps where it doesn't yield a lower value before stopping.

The authors use *Lasagne* for the CNN construction and training, *Pescador* is used to manage the data during training. In contrast we use *tensorflow*[3] highlevel api *keras* for construction and training.

3.3 Data Augmentation

The authors propose to make data augmentation over the audio signals prior to their conversion to the input feature space. They experiment with augmentation through time stretching, pitch shifting, Dynamic Range compression and Background noise. In this context it is important to choose the parameters of such augmentations so that the yielded data maintains the original data semantics.

The augmented dataset is constructed through the addition of the augmentations described in detail below.

- **Time Stretching:** Time stretch the sample by a factor a while maintaining the pitch unchanged. $a \in \{0.81, 0.93, 1.07, 1.23\}$.
- **Pitch shifting 1:** Change the pitch by b semitones while maintaining the time unchanged. $b \in \{2, -1, 1, 2\}$.
- **Pitch shifting 2:** Based on beneficial result in initial experiments the authors decide to extend the previous pitch shifting to larger values of b. $b \in \{-3.5, -2.5, 2.5, 3.5\}$.
- **Dynamic Range Compression:** Compression of the dynamic range of the signal. Three parameterizations are taken from the Dolby E Standard and one from the icecast online radio steaming server. The parameterizations are {*music standard, film standard, speech, radio*}.
- **Background Noise:** Mixing of the signal with four different urban background noise: {*street workers, street traffic, street people, park*}. The mixed signal is yielded as $z = (1 - x)x + wy$ were x, y are the signal and the background noise respectively. The trade-off parameter w is chosen for each mix from an uniform distribution in $[0.1, 0.5]$.

To apply this transformations to the data the authors use the MUDA library [4] which takes as inputs an audio file and a JAMS file [5], which is an annotation file format, and after a series of transformations outputs the deformed audio together with a JAMS file that contains the history of augmentations. This allows to replicate the augmented dataset from the augmented JAMS together with the original sounds. The authors kindly made them available on a github repository [2] together with the background noise scenes used.

[2] `https://github.com/justinsalamon/UrbanSound8K-JAMS`

3.4 Proposed Improvement

One clear conclusion from the authors results is that there is an special difficulty on the noise-like subset of of the dataset, as we can see on figure 3, this are the classes that don't exhibit clear large scale patterns as melodic-like classes do, then for these subset there is a greater importance in correctly identifying low-level patterns. We propose that separating the spectrogram image into higher and lower frequencies will leave the noise-like on the higher frequencies and in this way will allow a better learning of these patterns.

For this we propose to use the discrete wavelet transform (DWT), we further describe this concept on 6.2. The package *PyWavelet* is used to obtain the 2D single level wavelet decomposition. We get a one level DWT of the spectrogram image and we stack the approximation and details channel-wise. Furthermore this approach could have a general improvement by allowing the network to learn independent features for low and high frequencies more easily, as well as allowing it to be robust under background noise contamination in case we use such augmentation.

Additionally, the use of the wavelet transform can be thought as a biologically inspired feature extraction over the spectro-temporal characteristics, extracted in this case from the mel-spectrogram, as there are suggestions that the auditory cortex has spectro-temporal receptive fields and that such spectro-temporal receptive fields can be described by wavelets, see [8].

We use the same model of the authors but with a decreased filter size of $f_size = 3$, because as there is a downsample involved in the DWT, we want to maintain the "locallity" of the first layer features, furthermore due to the Maxpooling layer using the original filter size yields negative value dimensions on one point.

3.5 Evaluation

We evaluate the original model (SB-CNN) as well as the improved version (Wav-CNN) under three scenarios. The first two are the baseline scenarios with no augmentation, they are termed *baseline* and *baseline_multi*[3]. On *baseline* we only take the first 3 seconds of the signal and on *baseline_multi* we use the full signal as described by the authors. We separate this in order to assert if using all the data as described by the authors makes a significant difference in the achieved performance, as it implies a significant cost of "data streaming overhead" to the model training. Lastly, in the scenario termed *augmented* we test on the augmented dataset following the *baseline* scenario data handling.

The model are evaluated through 10-fold cross validation, which means training over 9 folds and leaving one for testing, and doing so for each of the 10 folds. From the 9 folds used for training one is chosen for validation in order to assert when the model is overfitting and training should stop. The dataset comes with predefined splits, this allow reproducibility and comparison of results across different studies over this dataset.

[3]J. Salomon was kind enough to confirm that for the non-augmented version of the model they used the same hyperparameters

To effectively observe the performance of the data we obtain box and wiskers plots for the test accuracy over all test folds for every model. Together with this we construct confusion matrices by summing the confusion matrices over every test fold, this allows us to identify what are the difficulties and common failures of every model.

4 Results

4.1 Original Results

The authors evaluate their model under two scenarios, with and without data augmentation. For this reproduction we only test the performance on their models, and don't reproduce the results of their baselines *SKM* and *PiczakCNN*.

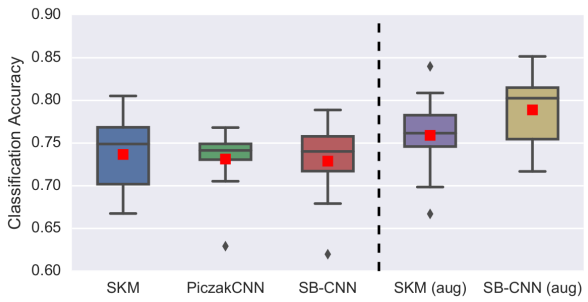

Figure 2: Original accuracy distributions across folds. Dashed line separates with and without augmentation results.

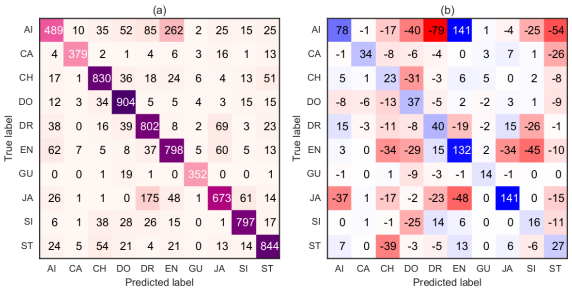

Figure 3: (a) Confusion matrix over all folds for augmented dataset. (b) Difference of confusion matrix for non-augmented case over the other.

We can see from the accuracy distributions that they improve significantly the performance of their model by using data augmentation, with an increase of the median close to 5% but with some added variance. We can see from figure 3(b) that there is great improvement inside noise-like mistakes.

4.2 Reproduction Results

As described on section 3.5 we tests the original model as well as the improved one on the scenarios *baseline*, *baseline_multi* and *augmented*. For easier understanding of the figures we will present bellow, the results from each of the scenarios will be named by concatenating to the model name an "*" for the *baseline_multi* scenario and " Aug" for the *augmented* scenario.

We must note that the *augmented* scenario was trained under a dataset constructed following *baseline* scenario, in consequence the reference should be done with respect to it and no to the *baseline_multi* scenario.

The accuracy distribution across folds for both models under all described scenarios are displayed on figure 4. We can see that the obtained results are significantly lower with respect to the original results, the distributions for the baseline scenarios have nearly a 10% and 15% lower median. The distribution over the *augmented* scenario has almost the same median as on the *baseline* scenario, but with an increased variance and has a median nearly 20% lower than on the original paper.

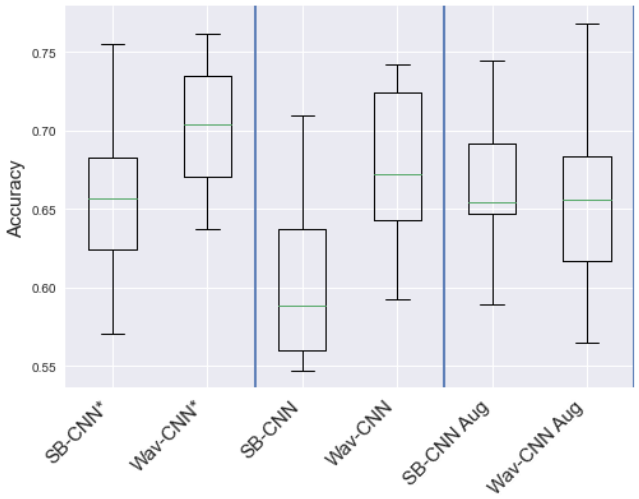

Figure 4: Accuracy distributions for SB-CNN and Wav-CNN under the previously described scenarios, divided by vertical lines. These results are separated with vertical lines over the scenarios *baseline_multi*, *baseline* and *augmented*.

We proposed to test under both baseline scenarios to see if there is a significant difference and we can see that there is a great difference in both models but more presently on SB-CNN which has a significant drop on the median and a great increase in variance.

Looking at the results on the *augmented* scenario of Wav-CNN we can see a very unexpected result, there is a minor decrease in accuracy and with greater variance.

Bellow we display a set of pairs of figures, where on the left figure there is a reference confusion matrix and on the right figure there is a difference of another model and/or scenario confusion matrix with respect to the reference. By showing these we intend to further pinpoint the effect of difference scenarios and different models over the set of classes we trained on. A confusion matrix represents the number of samples predicted as being from a class (vertical axis) that come from a certain class (horizontal axis). If the accuracy of the model is 100% it becomes a diagonal matrix with the total number of elements per class on it.

On Figure 5 we can observe the performance of the original model with training over the dataset as described by the authors. We can see that there are several noise-like classes that get confused for other noise-like classes, there are also many confusion on interpreting melodic-like samples as children playing, as well as several confusions of noise-like classes inferred as street music. From figure 6 we can see that there is no clear pattern of improvement of SB-CNN over SB-CNN* on the confusions among noise-like classes as there are gains and

losses, but there are two clear improvements over identifying jackhammer as drilling as well as overall improvement over detecting misidentifying as children playing and street music.

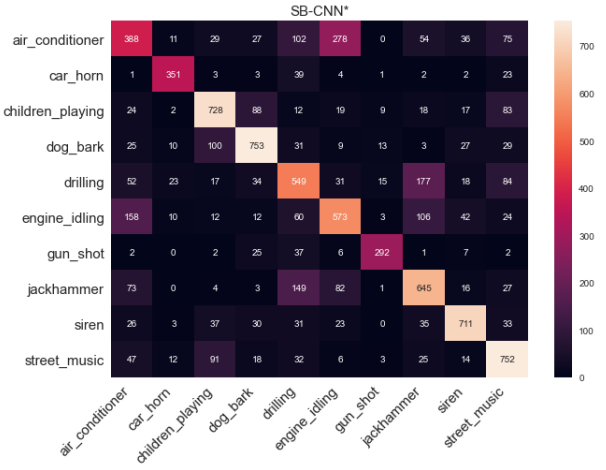

Figure 5: Confusion matrix of SB-CNN on the *baseline_multi* scenario.

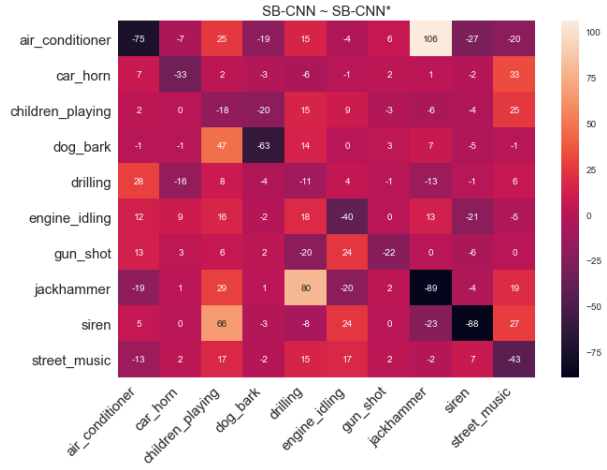

Figure 6: Difference of the confusion matrix of SB-CNN on the *baseline* scenario with respect to the same model on the *baseline_multi* scenario.

On figure 7 we have as a reference the model over the augmented dataset, most misconceptions are over noise-like classes, mainly between air conditioner, drillling, engine idling and jackhammer. Then on figure 8 we compare te previous reference to the non-augmented model, it suggests that augmentation lowered drilling mistakes at the expense of elevating the mistakes of detecting it as a jackhammer and elevated the number of detection of jackhammer as engine idling but decreased its performance over other classes, specially with drilling.

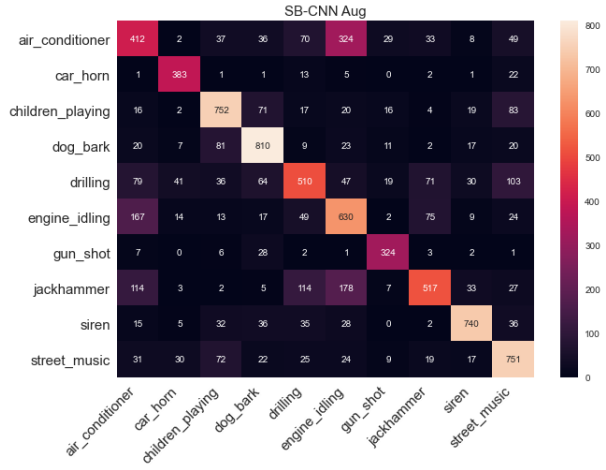

Figure 7: Confusion matrix of SB-CNN on the *augmented* scenario.

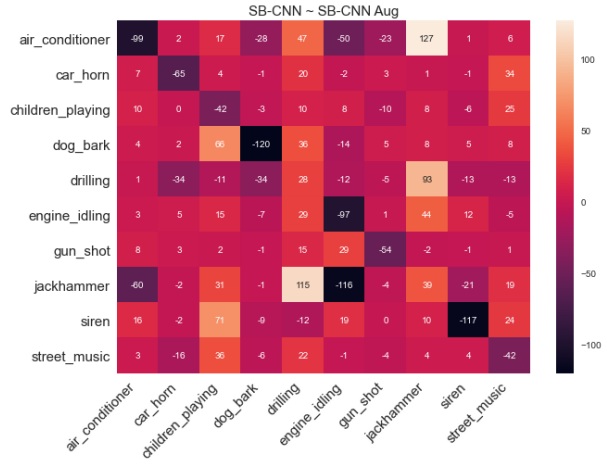

Figure 8: Difference of the confusion matrix of SB-CNN on the *baseline* scenario with respect to the same model on the *augmented* scenario.

From 9 we can see the performance of the improved model over the augmented dataset. From figure 10 we can see that the mistakes of both models follow a common pattern, so there was no improvement of performance over noise-like classes.

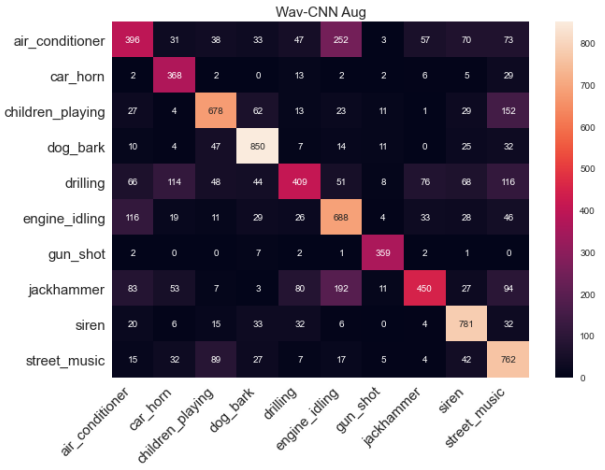

Figure 9: Confusion matrix of Wav-CNN on the *augmented* scenario.

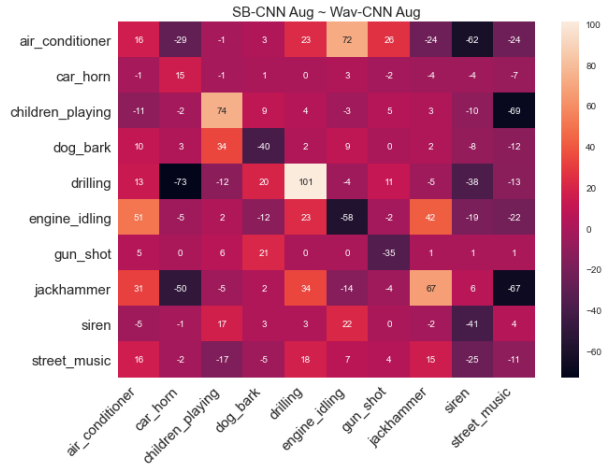

Figure 10: Difference of the confusion matrix of SB-CNN with respect to Wav-CNN on the *augmented* scenario.

From figure 11 and 12 we can see how this model yields similar results on both scenarios, there is a clear increase associated to every decrease and vice versa, for example overall air conditioner accuracy increases by decreasing mistakes of detecting them as engine idling, but in turn engine idling detection is worst with a greater number of mistakes.

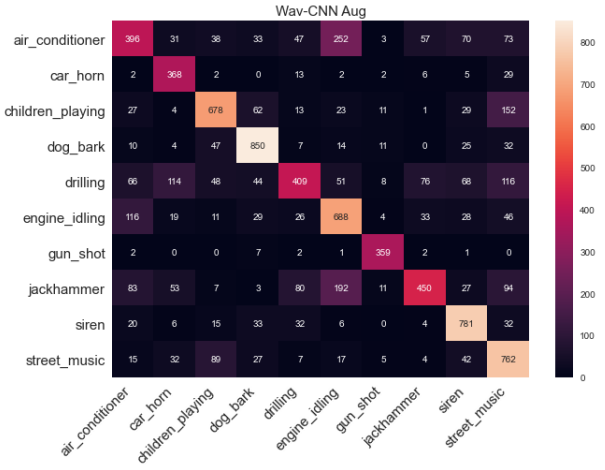

Figure 11: Confusion matrix of Wav-CNN on the *augmented* scenario.

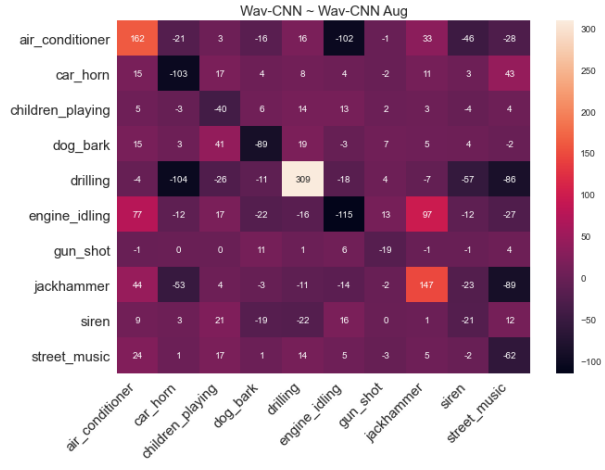

Figure 12: Difference of the confusion matrix of Wav-CNN on the *baseline* scenario with respect to the same model on the *augmented* scenario.

From figure 13 we see that the improved model over the *baseline_multi* has the same pattern of difficulties over noise-like classes we have seen before on other models and other scenarios. We can observe from figure 14 that it has lower mistakes of wrongly detecting as street music as well as significant fewer mistakes of detecting engine idling and jackhammer as air conditioner.

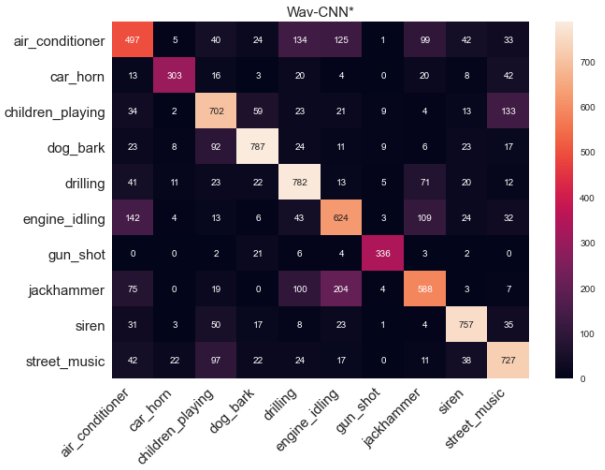

Figure 13: Confusion matrix of Wav-CNN on the *baseline_multi* scenario.

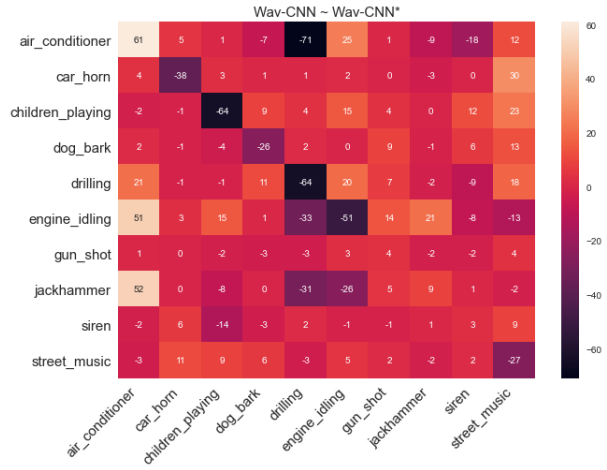

Figure 14: Difference of the confusion matrix of Wav-CNN on the *baseline* scenario with respect to the same model on the *baseline_multi* scenario.

All the used code for this results is available in this repository[4], there is a short description of the files there as well as the notebooks for each of the evaluated scenarios.

[4] `https://github.com/grudloff/Salomon2017Replication`

5 Conclusion

We weren't able to achieve the same results as the authors on the original paper, this suggests that the differences we had over their implementation play a huge role in improving performance.

One important difference is on the datastreaming on training, this is how we select data on each batch for training, which the authors implemented using one of *pescador*'s multiplexers[5].

Another important difference is on the selected data for training on each epoch, which as the authors stated they train over 1/8th of the total data, this is, all the possible excerpts from all data, which allows multiple instances from one sample to be used to train on a single epoch. In contrast we train over the whole dataset and select a random excerpt from each sample.

Finally it is possible that the use of early stopping is leading us to the detection of early plateau, which could be surpassed yielding better results with further training. This seems specially likely for the *augmented* scenario were most training stopped relatively early compared to the training of the baselines scenearios.

As we saw, there is a significative difference in the training over both baseline scenarios. Due to the intense time-consuming nature of their approach we decided to only test on the *baseline* scenario, but it is clear that a further testing over the *baseline_multi* scenario with data augmentation is needed.

One notable result from our proposed improvement, is that it achieves significantly better results over SB-CNN on the baseline scenarios. But the performance over the augmented dataset decreased, which is specially odd and counter intuitive, one possible explanation for this is the possibility of an early plateau mentioned early. Moreover we intended to assert the effect of the use of DWT as input space, in consequence we tried to change the model as little as possible, but clearly further improvements can be made by tuning the model hyperparameters.

[5]This was kindly answered by J. Salomon

6 Appendix

6.1 Mel Spectrogram.

The mel spectrogram is constructed by transforming the frequency axis of a spectrogram to the mel scale. The mel scale was first introduced by Stevens, Volkmann, and Newman in 1937 [6], and is based on the human perception of pitch. For this it has become the standard for machine audio perception, specially in speech analysis. The relation between the frequency and the mel scale is given by the following formula:

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

In general the mel spectrogram is computed by first computing the STFT, and mapping it to the mel scale by applying a filterbank of half-overlapping triangular filters which are equally spaced in the mel scale. As it is shown in the following figure.

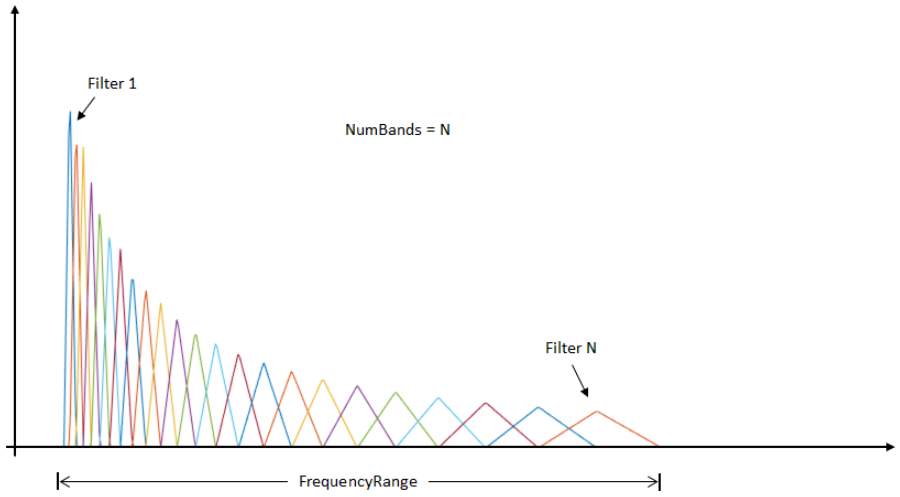

Figure 15: Mel filterbank with band normalization

Note that the filterbank displayed in the previous picture normalizes each of the filters to have the same area. This is not a standard but in general this is the default behaviour, and is the case for the default behaviour of *essentia* and *librosa*, as well as *matlab*'s implementation.

6.2 Discrete Wavelet Transform

The discrete wavelet transform can be thought as a filterbank, where on each decomposition level the high and low frequencies are separated by two filters in quadrature and subsequently are downsampled. Note that the low and high filters can be calculated from one another as they are in quadrature. These in turn can be obtained from the mother wavelet and father wavelet, also called wavelet function and scaling function, similarly they can be calculated from one another.

By definition the DWT is a representation of data by a basis formed by scalings and shifts of the mother and father wavelet, where each of them represents and "ideal signal" in a time-frequency band. In this sense even though the father and mother wavelets aren't necessary to calculate the DWT they can give us qualitative information of what kind of waveforms we are "observing" with the chosen basis.

The yielded low frequency representation is called approximations while the high frequency representation is called details, on each level these are calculated and the approximation is decomposed again to yield a subsequent level. After each level of decomposition the signal can be fully reconstructed from its current level approximation and all previous details. The process of deconstruction an reconstruction can be observed in the figure below for a one level decomposition.

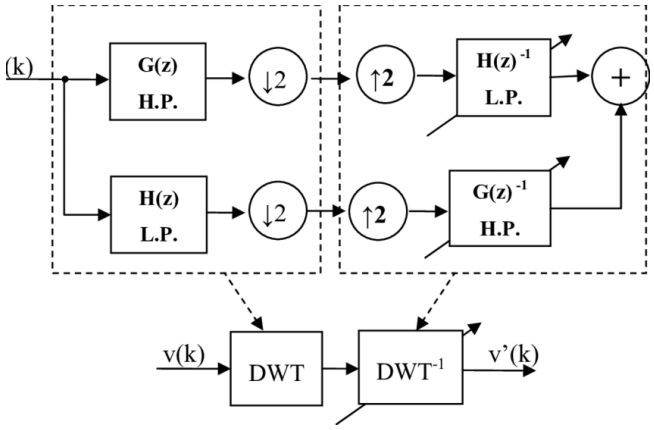

Figure 16: Discrete wavelet decomposition and reconstruction structure

For the 2D case, the DWT is applied colum-wise and then row-wise, although this is not how it is often implemented as there are more efficient ways. For a more in detail tutorial on the DWT check [9].

7 Acknowledgments

Specials thanks to Justin Salomon, one of the authors of the paper that was kind enough to answer some questions regarding their implementation.

References

[1] J. Salamon and J. P. Bello, "Deep Convolutional Neural Networks and Data Augmentation for Environmental Sound Classification," in IEEE Signal Processing Letters, vol. 24, no. 3, pp. 279-283, March 2017, doi: 10.1109/LSP.2017.2657381.

[2] J. Salamon, C. Jacoby and J. P. Bello, "A Dataset and Taxonomy for Urban Sound Research", 22nd ACM International Conference on Multimedia, Orlando USA, Nov. 2014

[3] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Rafal Jozefowicz, Yangqing Jia, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Mike Schuster, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng.
TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[4] McFee, B., Humphrey, E.J., and Bello, J.P. "A software framework for Musical Data Augmentation." 16th International Society for Music Information Retrival conference (ISMIR). 2015.
`https://github.com/bmcfee/muda`

[5] Eric J. Humphrey, Justin Salamon, Oriol Nieto, Jon Forsyth, Rachel M. Bittner, and Juan P. Bello, "JAMS: A JSON Annotated Music Specification for Reproducible MIR Research", Proceedings of the 15th International Conference on Music Information Retrieval, 2014. `https://github.com/marl/jams`

[6] Stevens, Stanley Smith; Volkmann; John & Newman, Edwin B. (1937). "A scale for the measurement of the psychological magnitude pitch". Journal of the Acoustical Society of America. 8 (3): 185–190

[7] Gregory R. Lee, Ralf Gommers, Filip Wasilewski, Kai Wohlfahrt, Aaron O'Leary (2019). PyWavelets: A Python package for wavelet analysis. Journal of Open Source Software, 4(36), 1237, `https://doi.org/10.21105/joss.01237`

[8] Qiu A, Schreiner CE, Escabí MA. Gabor analysis of auditory midbrain receptive fields: spectro-temporal and binaural composition. J Neurophysiol. 2003;90(1):456-476. doi:10.1152/jn.00851.2002

[9] T. K. Sarkar, C. Su, R. Adve, M. Salazar-Palma, L. Garcia-Castillo and R. R. Boix, "A tutorial on wavelets from an electrical engineering perspective. I. Discrete wavelet techniques," in IEEE Antennas and Propagation Magazine, vol. 40, no. 5, pp. 49-68, Oct. 1998, doi: 10.1109/74.735965.