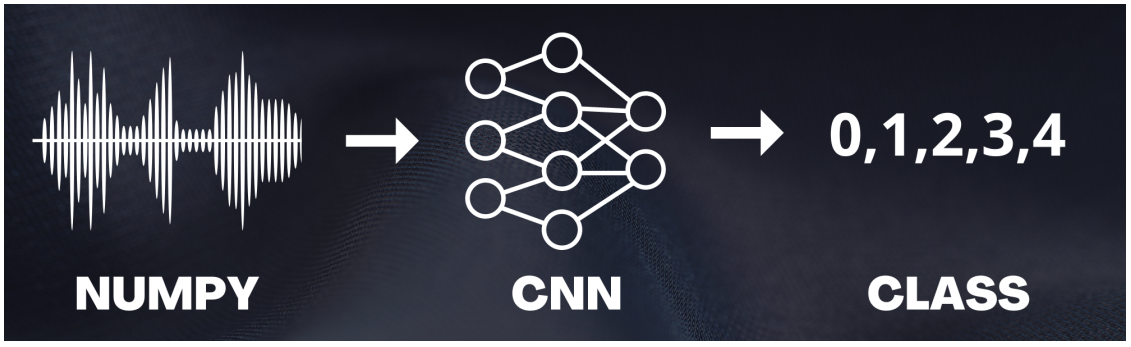

Audio Explorers challenge 2023

Pew Pew Sounds

2D CNN Sound Classifier

Authors:
Alexander Løvig Borg
Andreas Løvig Borg
Anton Sig Egholm

April 23, 2023

Contents

1 Introduction

Sound classification is a very popular application of machine learning for audio. Some of the many use cases are classifying genres of music, determining specific voice commands from the user, or transcribing sound files into text. It can also be used for determining who in the room is speaking by their vocal pattern, and even detecting their emotions by the inflection in their voice.

Sound classification can be done by, for example, detecting patterns in the spectograms of the audio and map them into features.

2 Process and Approach

When approaching the task, we tried various approaches with different machine learning models, to see which model performed the best in terms of prediction. This chapter will describe how well each model performed, and go into detail on Convolutional Neural Networks, as it was the model, that had the highest accuracy.

2.1 Comparison of Machine Learning Models

Table 1 presents a comparison of the performance of various machine learning models that we experimented with on the training data. It is important to note that not all models received equal attention or were developed with the same complexity.

Model	**Validation Accuracy**
Fully Connected Network (FCN)	0.6473
Logistic Regression (LR)	0.7733
Recurrent Neural Network (RNN)	0.6885
Random Forest Classifier (RFC)	0.7743
Convolutional Neural Network (CNN)	0.7929
2D Convolutional Neural Network (2D CNN)	0.8421

Table 1: Comparison of machine learning models and their validation accuracy using a 70% training and 30% validation split.

The results shown in Table 1 indicate that the 2D Convolutional Neural Network (2D CNN) model outperformed the other machine learning models in terms of validation accuracy. However, it is essential to recognise that this observation does not guarantee that the CNN is the best model for this task, as other models may not have been explored to their full potential. Further research and optimisation of these models could potentially reveal different outcomes.

2.2 Convolutional Neural Network

Convolutional Neural Networks (CNNs) have demonstrated their ability to excel in various computer vision tasks, such as image classification, and can be adapted for sound classification as well. Sound classification involves identifying the class or category of an audio signal, such as speech, music, or environmental sounds. In this report, we present a method to utilise a CNN for sound classification and demonstrate its implementation using a sample Python code.

The first step in sound classification is to convert the audio signal into a visual representation, such as a spectrogram. This representation captures the frequency and amplitude characteristics of the audio signal over time. The provided code assumes that the input data has already been preprocessed into spectrograms, which are loaded as NumPy arrays from files. The data is then split into training and validation sets, with 70% of the data used for training and 30% for validation.

The input data is normalised to a range between 0 and 1, as this often leads to better training convergence. The input dimensions are expanded to include the channel dimension, which is required for CNN input. The labels are converted to one-hot encoding to facilitate the use of categorical cross-entropy loss during training.

The CNN architecture used in this implementation consists of two convolutional layers, each followed by a max-pooling layer and dropout for regularisation. After the convolutional layers, the output is flattened and passed through a fully connected layer with 64 units and a dropout layer. Finally, a softmax activation function is applied to the output layer, which consists of as many units as there are classes in the dataset. The architecture can be seen in Figure 1.

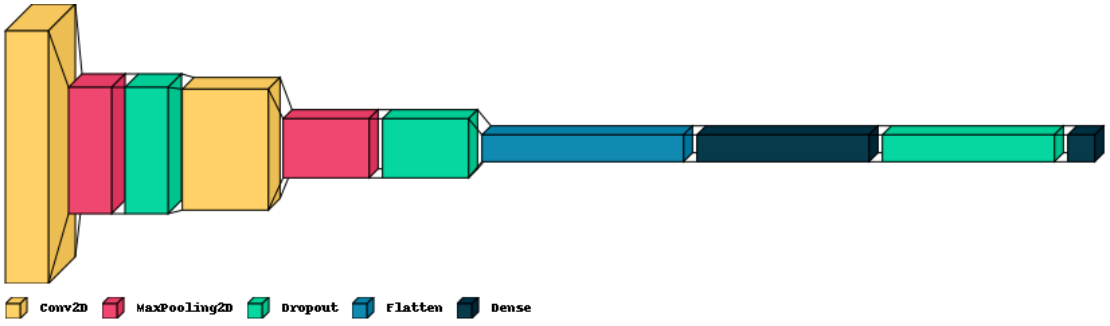

Figure 1: The basic architecture of the CNN model shown with Visual Keras.

The model is compiled using the Adam optimiser and categorical cross-entropy loss, and it is trained on the normalised training data for a specified number of epochs

with a batch size of 32. The model's performance is evaluated on the validation data, and the results, including the confusion matrix, are printed.

In conclusion, CNNs can be used for sound classification tasks by converting audio signals into visual representations, such as spectrograms, and designing a suitable CNN architecture for learning features from these representations. The provided code demonstrates an example of how to implement a simple CNN for sound classification using TensorFlow and Keras libraries.

3 Results

As mentioned, the code preprocesses the given spectrograms and splits them into training (70%) and validation (30%) sets. After normalization and one-hot encoding, a CNN architecture with two convolutional layers, max-pooling, dropout, and fully connected layers is trained on the data for 30 epochs with a batch size of 32.

The number of epochs significantly impacts the neural network training process. Too few epochs can result in underfitting, leading to high training and validation loss. Conversely, too many epochs can cause overfitting, with the training loss decreasing but the validation loss increasing. Monitoring training and validation loss helps identify the optimal number of epochs. Early stopping, which halts training when validation loss stops improving, prevents overfitting. In short, selecting the right number of epochs is crucial for achieving good performance while avoiding underfitting and overfitting. Figure 2 shows that the closest loss for training and validation was at around 25-30 epochs.

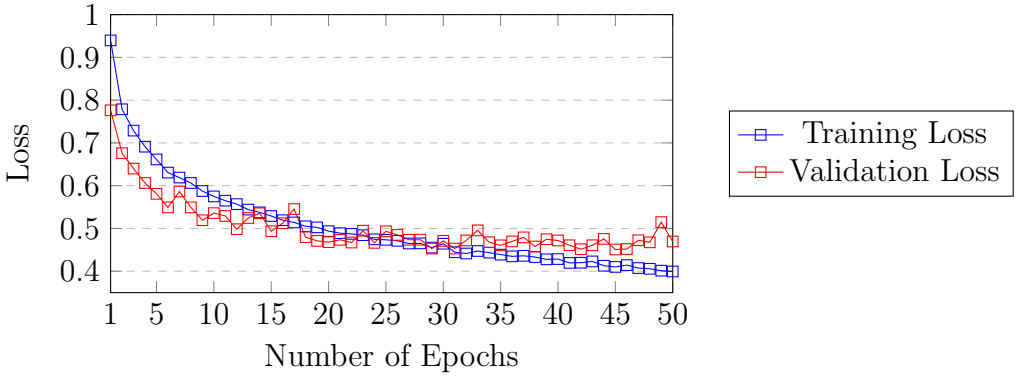

Figure 2: Training and Validation Loss Comparison

Based on the loss comparison we decided to go with 30 epochs which were then used to make predictions on the test data.

The labels as given in the challenge are: 0. Other, 1. Music, 2. Human voice, 3. Engine Sounds, 4. Alarm.

Results from the 2D CNN model with 30 epochs show a training loss of 0.4535, accuracy of 0.8319, validation loss of 0.4715, and validation accuracy of 0.8421. The confusion matrix can be seen in Table 2.

True Class	0	1	2	3	4
0	**3676**	225	151	264	8
1	214	**7841**	50	77	6
2	388	153	**880**	51	4
3	430	93	30	**780**	1
4	140	197	9	15	**184**

Predicted Class

Table 2: Confusion matrix for the CNN 2D model with 30% training size.

In conclusion, our code and results showcase the *potential* of CNNs for sound classification tasks using spectrogram representations. Additionally, the same confusion matrix, but with percentages, can be seen in Table 3

True Class	0	1	2	3	4
0	**86.0%**	5.3%	3.5%	6.2%	0.2%
1	2.6%	**95.9%**	0.6%	0.9%	0.1%
2	26.3%	10.4%	**59.6%**	3.5%	0.3%
3	32.6%	7.1%	2.3%	**59.3%**	0.1%
4	25.6%	36.1%	1.6%	2.7%	**33.7%**

Predicted Class

Table 3: Confusion matrix in percentages for the CNN 2D model with 30% training size.

4 Discussion

4.1 Accuracy

With the current performance and accuracy of the CNN, the model might not be suitable for commercial use, as the accuracy is low for predicting engine sounds and alarms. More specifically the number of true positives predictions for alarms were relatively low. This could be because engine sounds, alarms and "other" sounds have a similar acoustic features, which could be hard for the model to differentiate

between. Since alarms usually are associated with critical events, it is important to classify these correctly. Therefore this model should be improved at classifying these.

The performance of the model could be increased in general, if there was more data available to train on. Another way of increasing accuracy could be by adjusting weights of the different classes, to account for imbalance in the training data.

4.2 Computational Load

When deciding the type of algorithm to use for sound classification in hearing aids, the computational limits are important to consider. All algorithms have different time and space complexities for training and prediction[2][1]. The training complexity is not as important, as that can be done on a computer with more processing power. When the trained algorithm is imported to the hearing aid computer, the prediction time and space complexity can make a difference in performance.

The specific application of a machine learning algorithm together with a particular set of hardware and particular set of sound classes to predict can make it difficult to predict theoretical performance. While the theoretical computational load of prediction is important when deciding on a particular algorithm, it is also important to see how well it runs in practice. For this project, it would be ideal to be able to run our developed algorithm(s) on an actual OTICON hearing aid (or simulate the hardware), and the profile the algorithm to see how well it runs. Unfortunately with the way the OTICON challenge is set up, we are not able to live test our algorithm(s).

5 Conclusion

Based on the result of our training and validation, it can be concluded that the 2D CNN model we developed is effective in classifying sounds for Oticon's hearing aid. As mentioned in the discussion, further improvements could be made to the algorithm to ensure its viability in terms of classification accuracy. However, testing should be performed to determine the computational load on an actual OTICON hearing aid processor in a real-world setting.

Our Python code and model can all be seen on our GitHub:
https://github.com/andreaslborg/OticonChallenge2023

Additionally, the predictions for the *test.npy* file can found here:
https://github.com/andreaslborg/OticonChallenge2023/predictions.txt

References

[1] https://medium.com/analytics-vidhya/time-complexity-of-ml-models-4ec39fad2770.

[2] https://medium.com/datadailyread/computational-complexity-of-machine-learning-algorithms-16e7ffcafa7d.

[1] http://www.computerhistory.org/siliconengine/timeline/