

# Introduction

Hearing aids have come a long way in improving the quality of life for hearing-impaired individuals, using sophisticated algorithms to enhance hearing comprehension and reduce unwanted background noise. However, there is still a need to improve their performance in different sound environments. In this challenge, we design a sound environment classifier using a neural network that can label sound snippets as either “music”, “human voice”, “engine sounds”, “alarm” or “other”. The classifier uses relatively few parameters to accommodate the limited computational power of hearing aids. We also discuss how the classifier can be improved by implementing noise reduction, a different optimizer, or data preprocessing.

# Network Architecture

The model suggested in this report is a sequential convolutional network. The model has seven layers, including three convolutional layers, two 2D max-pooling layers, two dense layers, three dropout layers, and some batch normalization in between layers.

The focus has been on depth rather than width for the challenge at hand, since deep neural networks have proven to perform better in terms of accuracy. One might combine depth and breadth, but because the network is restricted to 500,000 parameters, it was determined to squeeze as much performance as possible out of a rather simple model.

Convolutional layers are frequently used in image classification applications because they excel at detecting spatial patterns in two-dimensional images. They can also be used for other forms of data with a spatial or temporal structure, such as audio or time series data. Each sound data sample in this project consists of 96 time frames and 32 frequency bands; each time-frequency point can be viewed as a pixel in a picture.

Three types of layers are added after each convolutional layer: batch normalization, 2D max-pooling, and random dropout. The goal of batch normalization is to increase model stability and speed up training by normalizing layer inputs by re-centering and re-scaling them.

Dropout layers were introduced after the initial two 2D max-pooling layers to combat overfitting, which was highly prevalent in this network. Other regularization approaches were tested, but only dropout layers were effective without compromising too much accuracy or loss.

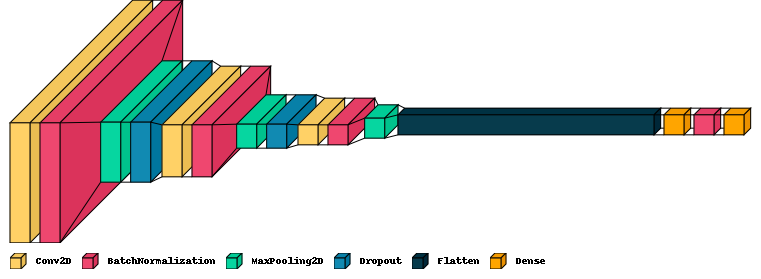


Fig 1: Visualization of the model

LeakyReLU was chosen as the activation function for the hidden layers since it performed the best among the tested activation functions. The output is passed through a softmax function, which is the typical function for multi-classification tasks. The sigmoid function can also be utilized; however, it involves adding thresholds and normalizing the network's outputs. If the network output requires a continuous value, a linear activation function might be used instead.

As the dataset contains multiple exclusive and one-hot-encoded classes, categorical loss entropy has been selected as the loss function in this network.

The Adam optimizer was picked due to its adaptive learning rate and momentum, which reduces the chance of the model getting stuck at a plateau or minimum. It also performs well on deep neural networks and large datasets.

# Results

For the first acceptable outcome, 317.365 parameters were used in the model, and an accuracy of 92% was achieved from the training with minimal overfitting. The subsequent evaluation of the test set proved to give an accuracy of 97%, which was better than anticipated given the performance during training. The confusion matrix below shows the accuracy of the predictions for each class.

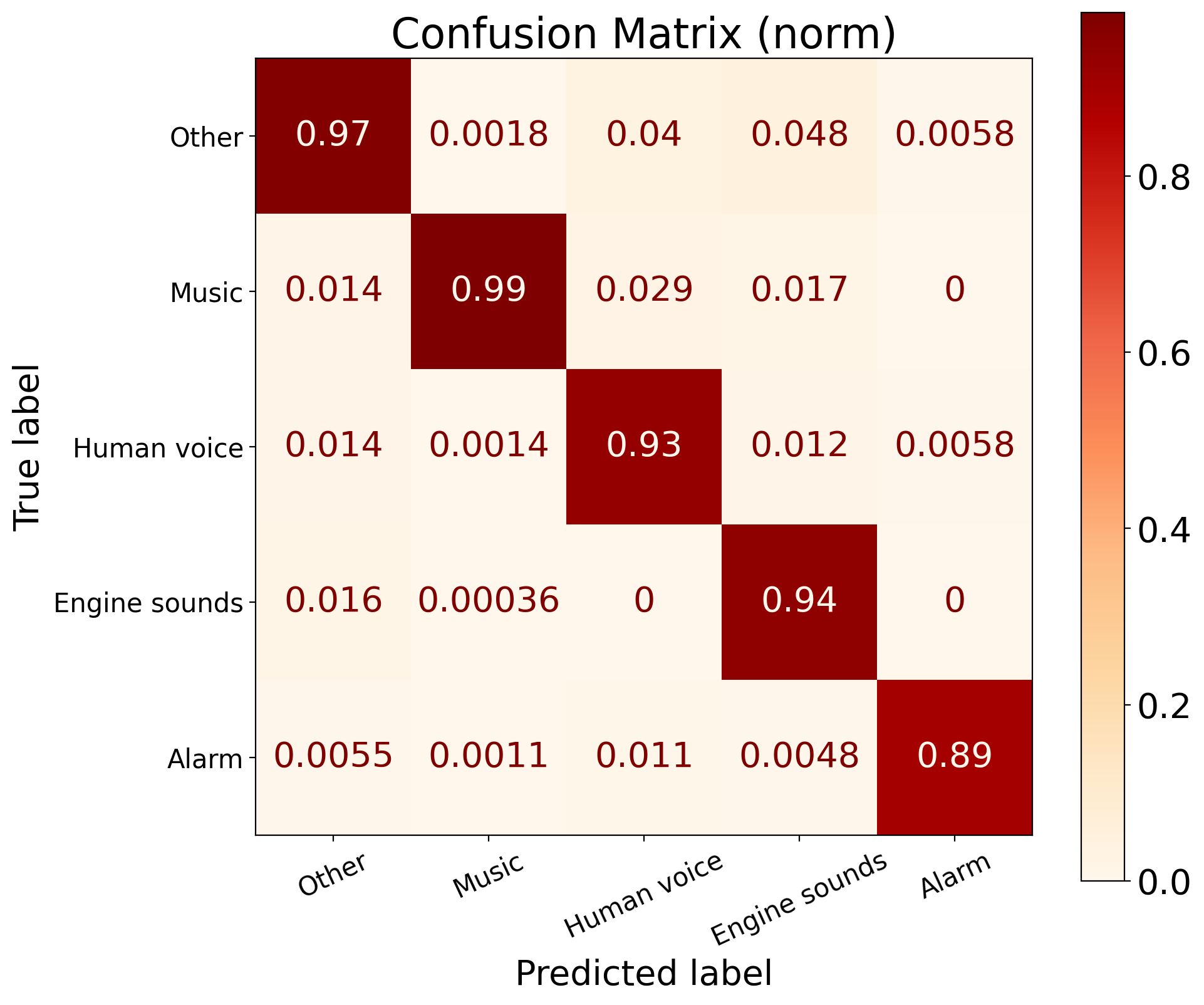


Fig 2: Normalized confusion matrix

The dataset split was done with an 80/10/10 ratio for training, evaluation, and testing.

The test set's class frequency is strongly skewed toward the “other” and “music” classes, with a frequency of 1500 samples for “other” and 2750 samples for “music”, whereas “human voice” and “engine sounds” only have around 500 samples, and “alarm” has approximately 250 samples.

This indicates a bias in the dataset sample distribution toward the first two classes, implying that performance may be improved by weighting the classes differently, if not just providing the dataset with additional examples of the final three classes, particularly the “alarm” class.

The confusion matrix indicates the model has the most trouble predicting warning sounds, owing to the frequency range and forms of alarms. This is evidenced by samples from the alert class being incorrectly anticipated to be from any of the other classes. Human voices can also seem like noise for morning alarms; therefore, a preprocessing filter may be useful. Further categorization of the “alarm” class might clear up the issue, but it would need even more samples. Only “other” and “human voice” are occasionally incorrectly classified as alarms, highlighting the previously mentioned reality that many alarm samples are polluted by human voices.

# Future Improvements

## Noise

Noise reduction is an essential step in audio processing, as it can be challenging for a model to classify data with excessive noise. Various methods exist for removing noise from data, each with its own strengths and weaknesses.

One approach involves mapping the frequencies of desired sounds and filtering out everything else. While this method requires significant effort to record and analyze desired sounds, the implementation is simple and computationally efficient with basic filters. Though there would probably be a relatively small range of frequencies left to filter out.

Another method involves removing all the frequencies outside the human audible frequency range (i.e., ~ 20 Hz–20 kHz). This is just as simple and efficient, but this has likely been done to the dataset before the handover.

Just as the model is taking inspiration from 2D image processing, noise reduction methods from image processing can also be utilized on the 2D spectrograms. From a visual inspection of some spectrograms, it seems Gaussian noise is the most prevalent, and as such, a Gaussian blur could be useful. But this would lead to a reduction in detail because edges become less defined and some data is lost in the process. It is also a heavier operation than the simple filter operations mentioned earlier.

While technically not noise reduction, image sharpening techniques could help make patterns in the data easier to spot by the model as they increase gradients such that, for example, smudged lines become sharper. This is also a relatively expensive operation compared to the filters.

## Choice of Optimizer

L2 norm regularization can be less effective for certain types of problems, such as those with sparse features or when the goal is feature selection. In these cases, L1 norm regularization can be more effective as it encourages sparsity in the weight matrix. If a noise filter is applied or noise is reduced by traditional vision algorithms, it could be beneficial to change to L1 norm regularization instead.

Adamax is a variation of the Adam optimizer that uses the infinity norm (maximum absolute value) of the gradient instead of the L2 norm. It can be more suitable for very high-dimensional problems or when the gradients are sparse, as it is less affected by noisy or sparse gradients. It is also less sensitive to the learning rate hyperparameter than the standard Adam optimizer. However, Adamax can converge more slowly than Adam for some problems, so it may require more tuning of the hyperparameters to achieve good performance.

# Experimenting with data preprocessing

To improve the model accuracy, it was tried to cut out outliers that would make the data normalization too skewered. This is also in an attempt to improve accuracy, since training on bad or noisy data creates a bad model. To do this, a z-score was calculated, and the outlier threshold was set to 3 standard deviations, where everything above past three standard deviations from the center was considered an outlier. This removed 8000 samples, which were deemed to be a bit much, but we tried to train on it regardless. This performed worse than without the z-score cleanup, so it was rolled back.

Data augmentation with a bit of translation and scaling was implemented to try to make the model more robust to try to bring down overfitting. This also proved to make the model perform worse than without. Adding data augmentation should help prepare the model for a real-world environment, where a spectrogram can be considered rather fixed, where simple augmentations can prove quite difficult to do if it should still resemble something real. There is probably still a reasonable way to apply data augmentation with good results to this problem, but that is beyond the scope of this assignment.

# Predictions of the premade test set

Exposing the classifier to the premade test set provided by Oticon granted quite an interesting result; A list containing only the predictions “4” and “1”, aka. only “music” and “alarm”. This caused quite a worry among the team, seeing as the model seemed to perform so well on the 80/10/10 split test set, which was previously unseen to the model. The model was then exposed to the self-defined test set to make predictions, which seemed to generate a much more promising result than the premade test set.

Therefore, the conclusion has been drawn that the Oticon team intended to test the robustness of the model by providing samples that only fell under the categories of one and four. This makes sense, as overfitting is a prevalent issue in the challenge at hand, and having only two labels represented among the samples would surely expose an overfitted model.

# Conclusion

This report presents a neural network model for classifying sound environments into five categories, namely “music”, “human voice”, “engine sounds”, “alarm”, and “other”. The model employs a sequential convolutional network with seven layers, including three convolutional layers, two max-pooling layers, and two dense layers, with batch normalization and dropout layers added after each convolutional layer to improve stability and prevent overfitting. The LeakyReLU activation function and categorical loss entropy were used in the model, and the Adam optimizer was employed due to its adaptability to deep neural networks and large datasets. After training on 90% of the dataset, the model showed an accuracy rate of 97% on the last, never before seen, 10% of the dataset. Future improvements suggested for the model include noise reduction techniques to improve classification accuracy and preprocessing filters to reduce the misclassification of alarms as human voices.

# Appendices

There are three appendices connected with this report.

1. A Jupyter Notebook (.ipynb) file containing all the code used for this project.  
   **hearing\_aids\_sound\_classifier.ipynb**
2. The sound stage classifier model (.h5 file), can be loaded with Tensorflow.  
   **317k\_LReLU\_97-percent.h5**
3. A text file (.txt) containing the predictions of the premade test set.  
   May not be viewable by Windows Notepad, but should work with WordPad or Notepad++.  
   **predictions.txt**