Proceedings of the 4th International Conference on Signal Processing and Machine Learning
DOI: 10.54254/2755-2721/79/20241308

Identify sound in raucous acoustic environment

Borui Zhuang[1,5,†], Yuchen Zhang[2,6,†], Zhongyu Wang[3,7,†], Zixuan Liu[4,*,†]

[1]Zhen Hai High School, Ninbo, 315200, China
[2]Department of Engineering, King's College London, London, WC2R 2LS, UK
[3]Department of Engineering, University of Warwick, Coventry, CV4 7AL, UK
[4]Wuhan British-China School, Wuhan, 430034, China

[5]brian1218@qq.com
[6]k21035856@kcl.ac.uk
[7]winsonwang2003@gmail.com
*Corresponding author email: 1655003692@qq.com
[†]All the authors contributed equally to this work and should be considered as co-first author.

Abstract. Due to 2023, over 200 million people worldwide are visually impaired. The needs of people with visual impairments receive scant attention in today's world. Most of them cannot walk independently on Crowded thoroughfares. There are still some challenges in developing assistive devices for the visually impaired. This paper focuses on a classification system within the earphone worn on the ear that can distinguish between different sounds and can be located by the Sharpless of the sound waves. The proposed method comprises two main modules: the first is to transfer the audio signals to Spectrograms, which is done in Python, and then a trained Convolutional Neural Network (CNN) is used in Matlab to identify different types of sounds. When tested in a real-life environment, this system proved useful and accurate in identifying dangerous signals. This innovation is intended to provide them with the optimal time to evacuate dangerous areas, ensuring their safety.

Keywords: Audio Signal Classification, CNN layer, Vision impairment, Python, Matlab.

1. Introduction

Nowadays, the visually impaired face many difficulties in their daily lives.[1] Out of all the five sense organs, the eyes are the most important, as 80 percent of what we perceive comes from our sense of sight [2]. According to World Health Organization (WHO), about 220 million individuals suffer from close or faraway vision impairment, and about 45 million are blind. The number of people affected is increasing with the growth of the world's population.

Early-onset severe vision impairment can restrict a child's verbal, emotional, social, and cognitive development, which can have long-term effects. Also, vision impairment impacts adults' life quality, such as social isolation, difficulty walking, and a higher risk of falls and fractures. The most critical inconvenience for people who are blind is that they have difficulty walking in busy and noisy streets because There are plenty of potential dangers on the roads. For example, dog attacks and gun shootings possibly happen on the street. Currently, in the event of a road emergency, visually impaired individuals

© 2024 The Authors. This is an open access article distributed under the terms of the Creative Commons Attribution License 4.0 (https://creativecommons.org/licenses/by/4.0/).

DOI: 10.54254/2755-2721/79/20241308

are limited to relying on others and evacuating. However, their response time is comparatively longer than that of sighted individuals. The gravest consequence is the potential loss of lives due to their initial inability to escape imminent danger.

In this paper, the technique of audio classification will be used to deal with this issue. By putting a particular signal into MATLAB and doing the training and recognition. Eventually, design an ear-worn device that can pick up danger signals from noisy environments and tell blind people where danger is occurring through vibrations.

2. Methods

2.1. A Method for Transforming Audio Data into STFT Images for Enhanced Classification

Firstly, we transform unprocessed audio signals into Short- time Fourier Transform (STFT) images which improves the performance of audio signal classification tasks.

The methodology is around using STFT. Short-time Fourier transform (STFT) is a sequence of Fourier transforms of a windowed signal. STFT provides the time-localized frequency in- formation for situations in which frequency components of a signal vary over time, whereas the standard Fourier transform provides the frequency information averaged over the entire signal time interval [3]. In practice, the audio signal is divided into short, overlapping Windows, and a Fourier transform is applied to each window to obtain the composite spectrum of the frequency components within these Windows.

We capture the audio features along the time series by windowing the entire audio signal. This iteration method extracts frequency features over time and adapts to the instability of precise audio signals.

In addition, the results of STFT will be converted to spectrograms. This is an important step involving 2-Dimentional visual representation. These spectrograms provide a useful way to convey the complex frequency-time dynamics of audio signals. The Y-axis of the spectrum represents frequency, and the X-axis represents time. The amplitude of the frequency at a particular time point is encoded as intensity or colour.

The conversion of spectrograms to image formats suitable for integration with image-based deep learning architectures is implemented. Images are systematically organized by assigning them to the directories corresponding to the specified category labels. The process creates directories automatically without human intervention.

The overall workflow proceeds in the following key stages: This process is initiated by the seamless integration of base

libraries tailored for audio processing (such as librosa) and base libraries for numerical computation (numpy), flexible data manipulation (pandas), and efficient data visualization (matplotlib).

Then, the audio data was analyzed by Short Time Fourier Transform (STFT). This stage involves splitting the audio data into concise, overlapping Windows and applying a Fourier transform to extract key frequency components. This approach reveals the complexity of the spectrogram of each segment while able to have a detailed examination of the frequency dynamics within the signal.

Time shifts continued to be closely monitored during STFT analysis. A series of ongoing Fourier transforms are systematically computed as the window moves continuously over the audio signal. This dynamic analysis is a powerful tool for determining temporal patterns and frequency fluctuations.

The next stage is generating spectra. These spectra constitute together to have a visual representation that effectively shows the interaction between frequency and time. The Y-axis represents the frequency and the X-axis represents the time, and the colour intensity represents the magnitude of a particular frequency component. Spectrograms provide an intuitive and insightful way to encapsulate complex audio information.

In addition, the naming scheme for these images is the same as the original audio file names, with support for including the “Portable Network Graphics (PNG)” extension for enhanced clarity and discrimination.

DOI: 10.54254/2755-2721/79/20241308

In summary, our code ensures that we have efficient storage and preserve information when transforming audio data into spectral images. This method combines the analysis of audio signals by using image-based deep learning technology and builds a foundation for enhancing audio classification. We will modify the window parameters and investigate advanced image processing techniques in future.

2.2. CNN model and model training for classification

In our research project, we build a CNN model with 8 different layers and select the method called stochastic gradient descent as the iteration method. [4,5] Then we use 2 training methods for classification. One Is called K-fold cross-validation and the second one is the general method. The aim of all these is to load in the image train the model and do the classification.

For the layers of the model:

Image-input layer: This layer specifies the size of the image. We select the size 400*1000*3 which is exactly the same as the PNG image we preserve from Python.

Convolution layer: This layer filters the input and extracts features of the image by detecting various patterns. After several experiments, we select the layer with 32 filters of size 3*3.

Batch Normalization layer: It aims to normalize the activations of the previous layer which improves training stability and convergence speed.

ReLU Layer: This activation layer introduces nonlinearity to the network by applying the Rectified Linear Unit (ReLU) activation function to the output of the previous layer.

Max Pooling Layer: This layer reduces the spatial dimensions of the feature map using max pooling with a 2x2 window and a stride of 2.

Fully Connected Layer: This layer is fully connected and maps the extracted features to a size-2 vector which is suitable for classification.

Softmax Layer: This layer converts the output of the previous fully connected layer into class probabilities using the softmax function.

Classification Layer: The final layer is a classification layer that performs the classification task and calculates the loss during training.

The k-fold cross-validation is recommended for small data as it divided the data equally into k subsets. Then select 1 sub- set for validation and k-1 subsets for training. Switch different subsets for validation and run for k times. Average the accuracy in order to get a more accurate result.[6]

The general method is to split the data into 3 data groups: Training, Testing and Validation group. Then use the training group trains the model and then tested by the other 2 groups, this method often has a great performance with large data.

3. Results

During our research, we use Python to convert audio into visual representations, which is useful for audio classification tasks and then we carried out training, testing, and validation in MATLAB. Two different methods called Simple Train-Test- Validation Splits and K-fold Cross-Validation are used. [6] Below is a summary of the accuracy of both methods.

Table 1. Train-Test-Validation method accuracy of the testing set

Method	Accuracy
Train-Test-Validation Split with 170 data	70.558
Train-Test-Validation Split with 1500 data	83.553

Table 2. Train-Test-Validation method accuracy of the validation set

Method	Accuracy
Train-Test-Validation Split with 170 data	64.706
Train-Test-Validation Split with 170 data	82.353

DOI: 10.54254/2755-2721/79/20241308

It is obvious that as the number of data increases, the accuracy increases in both data sets. In addition, there are several reasons why the accuracy in the testing set is higher than that in the validation set.

1:Random Variation: Even if we split our data into training, testing, and validation sets with a fixed seed for randomness, there are still some levels of randomness involved.

2:Data Splitting: The way we split your data into training, testing, and validation sets can affect the performance of each set. If, by chance, our testing set contains examples that are

more similar to the training set, the model could perform better on the test set.

3:Optimization Process: While the optimization process aims to minimize the loss function, it's an iterative process that depends on various factors such as initialization, learning rate, and mini-batch sampling. These factors can lead to small differences in convergence between the validation and testing sets. Next, we put the trained data into another model (k-fold cross- validation), and the following diagram shows the result.

Table 3. Accuracy comparison with 2 methods when the number of data is 170

Method	Accuracy
K-fold cross-validation with 170 data	70.558
K-fold cross-validation with 170 data	91.667

The result given by k-fold cross-validation with 170 data is much higher than its counterpart, and it is also about 10 percent higher than using the first method with 1500 data. We also try to put 1500 data into this model to do training, unfortunately, it exceeds MATLAB's capacity and it cannot give us feedback, that's a pity.

Overall, we find that when we only have a small number of data, the top priority is to use k-fold cross-validation since it will give a much higher accuracy compared with Simple Train- Test-Validation Splits. However, when it comes to a large number of data, this method is quite time-consuming and expensive. Therefore, it is better to use Train-Test-Validation Splits as it also can give relatively high accuracy.

4. Discussion

Our project is focused on signal filtering, model training and classification. We have succeeded in high precision of identifying important audio. We aim to finish audio locating, voice broadcasting and software design in future or cooperate with other research groups in order to provide a full system that helps people with vision impairment in another way.[7]

5. Summary and conclusions

The proposed method of audio classification can identify dangerous signals with high accuracy, at about 66 percent.[4-6] So, this proved that this system could help the visually impaired to know where the danger is in the first place in a busy street in real-life. In this situation, we plan to develop a system that can distinguish between different audio and detect the surrounding environment, which can be notified by the device's voice broad- cast to help the visually impaired walk independently and take more effective measures when there is an emergency. We plan to combine our method with a camera called Third Eye, pro- posed by four other researchers and finally realize our complete idea.[7]

Acknowledgements

Yuchen Zhang, Borui Zhuang, Zhongyu Wang and Zixuan Liu contributed equally to this work and should be considered co-first authors. Throughout the writing of this dissertation, We would like to thank Professor Neal Bangerter for providing helpful suggestions and leading on the methodology. Your insightful feedback pushed us to sharpen and widen our thinking and brought our work to a higher level. We also appreciate the valuable feedback from teaching assistants Taylor and Jason. Also thanks for the coding assistant.

DOI: 10.54254/2755-2721/79/20241308

Appendix

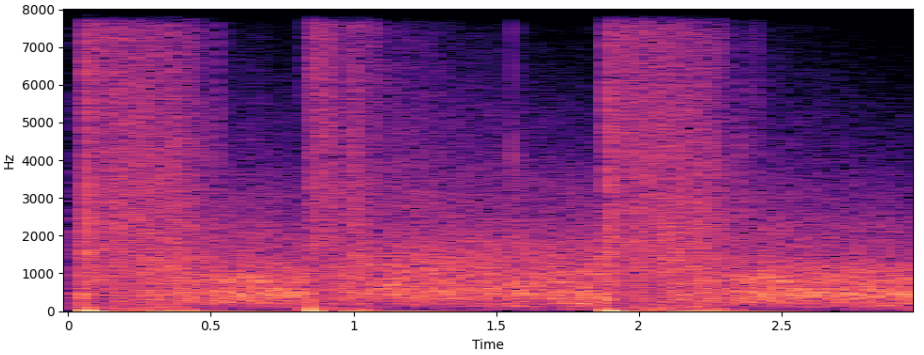

Figure 1. Spectrogram of gunshot audio signal

```
>> sample_code1
Number of training images: 136
Number of testing images: 17
Number of validation images: 17
Training on single CPU.
Initializing input data normalization.
|=========================================================================================|
|  Epoch  |  Iteration  |  Time Elapsed  |  Mini-batch  |  Mini-batch  |  Base Learning  |
|         |             |   (hh:mm:ss)   |   Accuracy   |     Loss     |      Rate       |
|=========================================================================================|
|       1 |           1 |       00:00:02 |       75.00% |       0.4834 |          0.0010 |
|       7 |          50 |       00:02:12 |       93.75% |       0.2895 |          0.0010 |
|      10 |          80 |       00:03:19 |       75.00% |       0.4038 |          0.0010 |
|=========================================================================================|
Training finished: Max epochs completed.
Accuracy on testing set: 0.70588
Accuracy on validation set: 0.64706
```

Figure 2. Train-Test-Validation method of 170 data

```
>> sample_code1
Number of training images: 1222
Number of testing images: 152
Number of validation images: 153
Training on single CPU.
Initializing input data normalization.
|=========================================================================================|
|  Epoch  |  Iteration  |  Time Elapsed  |  Mini-batch  |  Mini-batch  |  Base Learning  |
|         |             |   (hh:mm:ss)   |   Accuracy   |     Loss     |      Rate       |
|=========================================================================================|
|       1 |           1 |       00:00:02 |       31.25% |       1.8513 |          0.0010 |
|       1 |          50 |       00:01:46 |       50.00% |       0.6932 |          0.0010 |
|       2 |         100 |       00:03:30 |       81.25% |       0.3578 |          0.0010 |
|       2 |         150 |       00:05:15 |       87.50% |       0.3404 |          0.0010 |
|       3 |         200 |       00:06:59 |       93.75% |       0.2347 |          0.0010 |
|       4 |         250 |       00:08:44 |       87.50% |       0.3004 |          0.0010 |
|       4 |         300 |       00:10:29 |       68.75% |       0.6175 |          0.0010 |
|       5 |         350 |       00:12:14 |       93.75% |       0.1002 |          0.0010 |
|       6 |         400 |       00:13:59 |       87.50% |       0.3158 |          0.0010 |
|       6 |         450 |       00:15:44 |       93.75% |       0.1470 |          0.0010 |
|       7 |         500 |       00:17:28 |       81.25% |       0.2902 |          0.0010 |
|       8 |         550 |       00:19:13 |       87.50% |       0.3745 |          0.0010 |
|       8 |         600 |       00:21:00 |       87.50% |       0.2841 |          0.0010 |
|       9 |         650 |       00:22:53 |       87.50% |       0.5693 |          0.0010 |
|      10 |         700 |       00:24:46 |      100.00% |       0.1163 |          0.0010 |
|      10 |         750 |       00:26:38 |      100.00% |       0.0345 |          0.0010 |
|      10 |         760 |       00:27:00 |       87.50% |       0.1675 |          0.0010 |
|=========================================================================================|
Training finished: Max epochs completed.
Accuracy on testing set: 0.83553
Accuracy on validation set: 0.82353
```

Figure 3. Train-Test-Validation method of 1500 data

DOI: 10.54254/2755-2721/79/20241308

```
>> testcode
Training on single CPU.
Initializing input data normalization.
|========================================================================================|
|  Epoch  |  Iteration  |  Time Elapsed  |  Mini-batch  |  Mini-batch  |  Base Learning  |
|         |             |   (hh:mm:ss)   |   Accuracy   |     Loss     |      Rate       |
|========================================================================================|
|       1 |           1 |       00:00:22 |       42.19% |       1.7838 |          0.0010 |
|       8 |           8 |       00:02:55 |       91.41% |      22.4419 |          0.0010 |
|========================================================================================|
Training finished: Max epochs completed.
Training on single CPU.
Initializing input data normalization.
|========================================================================================|
|  Epoch  |  Iteration  |  Time Elapsed  |  Mini-batch  |  Mini-batch  |  Base Learning  |
|         |             |   (hh:mm:ss)   |   Accuracy   |     Loss     |      Rate       |
|========================================================================================|
|       1 |           1 |       00:00:26 |       37.50% |       0.9969 |          0.0010 |
|       8 |           8 |       00:02:50 |       91.41% |      46.9998 |          0.0010 |
|========================================================================================|
Training finished: Max epochs completed.
Training on single CPU.
Initializing input data normalization.
|========================================================================================|
|  Epoch  |  Iteration  |  Time Elapsed  |  Mini-batch  |  Mini-batch  |  Base Learning  |
|         |             |   (hh:mm:ss)   |   Accuracy   |     Loss     |      Rate       |
|========================================================================================|
|       1 |           1 |       00:00:28 |       39.06% |       1.7723 |          0.0010 |
|       8 |           8 |       00:03:07 |       91.41% |      35.6676 |          0.0010 |
|========================================================================================|
Training finished: Max epochs completed.
Training on single CPU.
Initializing input data normalization.
|========================================================================================|
|  Epoch  |  Iteration  |  Time Elapsed  |  Mini-batch  |  Mini-batch  |  Base Learning  |
|         |             |   (hh:mm:ss)   |   Accuracy   |     Loss     |      Rate       |
|========================================================================================|
|       1 |           1 |       00:00:25 |       39.84% |       1.2568 |          0.0010 |
|       8 |           8 |       00:02:51 |       90.62% |      33.3025 |          0.0010 |
|========================================================================================|
Training finished: Max epochs completed.
Average accuracy with k-fold cross-validation: 0.91667
```

Figure 4. k-fold cross-validation method of 170 data

References

[1] C. L. Themann and E. A. Masterson, “Occupational noise exposure: A review of its effects, epidemiology, and impact with recommendations for reducing its burden,” Journal of the Acoustical Society of America, vol. 146, no. 5, pp. 3879–3905, Nov. 2019, doi: 10.1121/1.5134465.

[2] G. Licitra, M. Bolognese, C. Chiari, S. Carpita, and L. Fredi- anelli, “Noise Source Predominance Map: a new representation for strategic noise maps,” Noise Mapping, vol. 9, no. 1, pp. 269–279, Jan. 2022, doi: 10.1515/noise-2022-0163.

[3] Nasser Kehtarnavaz, ‘Digital Signal Processing System Design (Second Edition)’, LabVIEW-Based Hybrid Program- ming, 2008, Pages 175-181, 183-196

[4] “Stochastic gradient descent - Cornell University Compu- tational Optimization Open Textbook - Optimization Wiki.” https://optimization.cbe.cornell.edu/index.php?title=Stochastic gradient descent

[5] T. O¨zseven, “Investigation of the eff ectiveness of time- frequency domain images and acoustic features in urban sound classification,” Applied Acoustics, vol. 211, p. 109564, Aug. 2023, doi: 10.1016/j.apacoust.2023.109564.

[6] J. Brownlee, “A Gentle Introduction to k-fold Cross- Validation,” MachineLearningMastery.com, Aug. 2020, [On- line]. Available: https://machinelearningmastery.com/k-fold- cross-validation/

[7] “Zero-Shot Audio Classification Based On Class Label Em- beddings,” IEEE Conference Publication — IEEE Xplore, Oct. 01, 2019. https://ieeexplore.ieee.org/document/8937283