Journal of Artificial Intelligence and Engineering Applications

Website: https://ioinformatic.org/

June 2026. Vol. 5. No. 3; e-ISSN: 2808-4519

Implementation of Convolutional Neural Network for Emergency Sound Detection for Hearing-Impaired Individuals on Android

Muhammad Akram Fais[1]*, Insan Taufik[2], Mansur AS[3], Debi Yandra Niska[4], Hanna Dewi Marina Hutabarat[5]

[1,2,3,4,5]*State University of Medan*
mhdakramfais@mhs.unimed.ac.id[1*], insantaufik@unimed.ac.id[2], asmansur@unimed.ac.id[3], debiyandraniska@unimed.ac.id[4], hanahutabarat@unimed.ac.id[5]

Abstract

Hearing impairment is a condition characterized by partial or total loss of hearing ability, which may occur congenitally or be caused by factors such as injury, disease, or prolonged exposure to excessive noise. This study aims to develop an Android-based emergency sound detection system using the Convolutional Neural Network (CNN) method. The research workflow includes problem identification, data collection, data preprocessing, CNN model training, model evaluation, Android application development, and system testing. Experimental results show that the best-performing model achieved an overall accuracy of 93%. The trained model was then implemented into an Android application to enable real-time sound classification and to provide visual notifications when emergency sounds are detected. The evaluation results indicate that the CNN model is capable of accurately classifying emergency sounds and operates effectively on Android devices.

Keywords: *Deaf; Emergency Sound; Machine Learning; Convolutional Neural Network; Android*

1. Introduction

Hearing impairment is a condition characterized by the partial or complete loss of hearing ability, which may occur congenitally or be caused by various factors such as injury, disease, or prolonged exposure to excessive noise [1]. According to data from the World Health Organization, approximately 1.5 billion people worldwide experience some degree of hearing loss, representing around 20% of the global population [2]. Of this number, 430 million individuals suffer from disabling hearing loss (deafness). In Indonesia, data from the Central Bureau of Statistics indicate that 255,161 individuals are completely unable to hear.

One of the primary challenges faced by individuals with hearing impairment is the difficulty in recognizing sounds, particularly emergency sounds, which can have serious implications for their safety in critical situations. Most emergency information is conveyed through auditory signals; thus, the inability to perceive warning sounds significantly increases their vulnerability in such scenarios. Consequently, individuals with hearing impairment may be unable to respond promptly to hazardous situations. Therefore, there is a pressing need for technological solutions capable of converting emergency auditory information into visual representations.

The Android platform enables developers to design and develop applications tailored to specific user needs, including emergency sound detection for individuals with hearing impairment. Its open-source nature, support for visual accessibility features and notifications, as well as its capability to integrate with machine learning technologies, make Android a suitable platform for the application proposed in this study [3]. One of the machine learning approaches proven effective in sound classification is the Convolutional Neural Network (CNN), a deep learning architecture capable of automatically extracting features from spectral representations of audio signals without requiring manual feature engineering [4]. The ability of CNNs to handle complex acoustic variations, such as differences in intensity and background noise, makes them superior to conventional methods [5].

Several previous studies have demonstrated the effectiveness of CNNs in sound classification. Rafliansyah et al. successfully classified the sounds of wind instruments using a combination of CNNs and Mel-Frequency Cepstrum Coefficients (MFCCs), achieving a maximum accuracy of 84% [6]. Wang & Goldsztein demonstrated that audio representations in the form of spectrograms can be analyzed using CNNs with high accuracy [4]. Sehgal & Kehtarnavaz developed a CNN-based smartphone application for real-time sound activity detection on hearing aids, with results that surpassed previous detection methods [7]. However, research specifically integrating CNN models into Android applications for emergency sound detection for the deaf remains limited.

Based on this urgency, this study aims to design and implement a CNN model capable of detecting and classifying various types of emergency sounds, as well as to develop an Android-based application integrated with the model to provide real-time notifications to

individuals with hearing impairment during emergency situations. This research is expected to contribute significantly to enhancing independence and ensuring safety for individuals with hearing impairment through the application of deep learning technology integrated with the Android platform.

2. Literature Review

2.1. Hearing Impairment

Hearing impairment is a condition characterized by partial or complete loss of auditory function, resulting in an individual's inability to perceive stimuli through the sense of hearing [8]. This condition may be congenital (present at birth) or acquired due to various factors such as aging, prolonged exposure to noise, disease, physical trauma, or chemical agents [9]. Based on the degree of hearing loss, hearing impairment is classified into five categories: mild (20–30 dB), marginal (30–40 dB), moderate (40–60 dB), severe (60–75 dB), and profound (>75 dB). The mild to moderate categories are generally classified as hard of hearing, whereas the severe and profound categories are classified as deaf [8].

2.2. Emergency Sounds

Emergency sounds constitute a critical component of early warning systems designed to rapidly convey safety instructions and evacuation guidance to individuals in life-threatening situations. According to the Indonesian National Standard (SNI) 03-6574-2001, hazard warning systems function to provide clear and accurate instructions in order to prevent panic, which may otherwise exacerbate safety risks [10]. Examples of emergency sounds include fire sirens, disaster alarms, cries for help, and explosion sounds.

From an acoustic perspective, emergency sounds are designed with distinctive frequency, intensity, and temporal patterns to ensure rapid recognition, even in environments with high levels of background noise [11]. ISO 7731:2003 specifies three primary criteria that emergency sounds must fulfill: (1) audibility, referring to the ability of the sound to be perceived across the intended area despite background noise; (2) distinctiveness, defined as a unique and easily distinguishable sound pattern compared to other sounds; and (3) intelligibility, referring to the clarity of the conveyed message, particularly in panic situations[12].

2.3. Spectogram

A spectrogram is a visual representation of the frequency spectrum of an audio signal as it varies over time [13]. This representation is generated using the Short-Time Fourier Transform (STFT), in which a one-dimensional audio signal is segmented into short time intervals, and each segment is analyzed to extract its frequency components. The resulting analysis is visualized as a two-dimensional matrix, where the horizontal axis represents time, the vertical axis represents frequency, and the color intensity corresponds to the amplitude at each time–frequency point. The mathematical formulation of the STFT is expressed as follows:

$$X(\tau, \omega) = \int_{-\infty}^{\infty} x(t)\, w(t-\tau) e^{-j\omega\tau} dt \quad (1)$$

where $w(t-\tau)$ denotes the window function used to segment and filter the signal. The selection of the window length must be carefully adapted to the characteristics of the signal, as it directly influences the trade-off between time resolution and frequency resolution (Rizal et al., 2016). The primary advantage of the spectrogram over conventional Fourier analysis lies in its ability to simultaneously represent both temporal and frequency information [14].

2.4. Convolutional Neural Network

Convolutional Neural Networks (CNNs), also known as convolutional networks, are a specialized type of neural network designed to process data with a matrix-like topology. Examples include image data, which can be represented as a 2D pixel matrix, and time-series data, which can be considered a 1D matrix sampled at specific time intervals. CNNs utilize convolution operations as the basis for extracting features from input data, making them highly effective for tasks such as image recognition, object detection, and time-series data analysis [15].

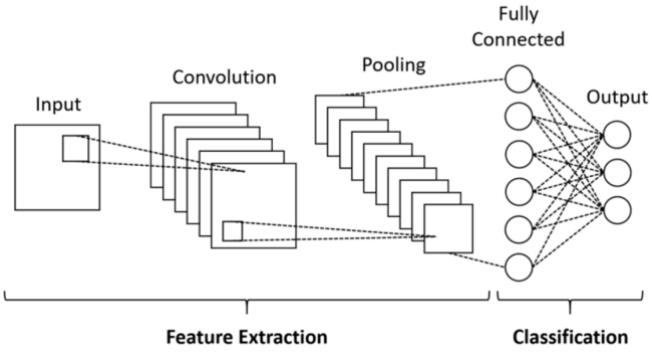


Fig. 1: Convolutional Neural Network Architecture

2.5. Android

Android is a mobile operating system based on the Linux kernel, developed by Google following its acquisition of Android, Inc. in 2005 [16]. Its open-source nature makes Android a flexible and highly customizable platform that can be tailored to specific user needs, including the development of accessibility applications for individuals with disabilities. Android application development is commonly carried out using Java or Kotlin, utilizing the Android Software Development Kit (SDK), which provides a comprehensive set of development tools, including a debugger, software libraries, and a device emulator [17]. The compiled code, along with all application resources, is packaged into an Android Package (APK) format, which serves as the installation package for Android devices.

3. Research Methodology

The research workflow can be observed in the flowchart presented below.

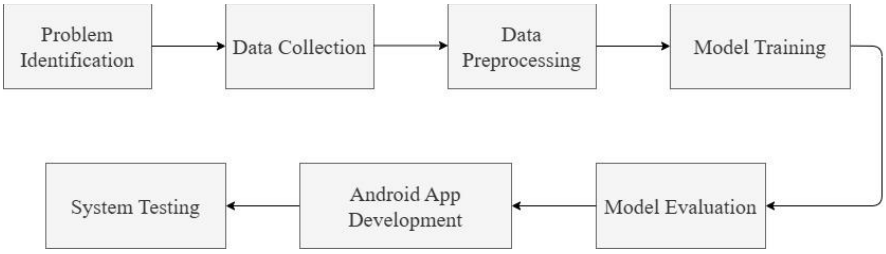


Fig. 2: Research Workflow

The following section describes each stage of the research process.

3.1. Problem Identification

Problem identification constitutes the initial stage of the research, aimed at understanding the issues to be addressed and determining appropriate solutions. This process involves analyzing the types of emergency sounds to be detected, the challenges encountered, and environmental conditions that may affect the recognition of such sounds. In this study, the identified problem relates to the difficulty experienced by individuals with hearing impairment in recognizing emergency sounds in their surroundings, which may pose serious safety risks if not promptly detected.

3.2. Data Collection

The data collection process was conducted to obtain emergency sound data for model training, validation, and testing. The collected audio samples include ambulance sirens, vehicle horns, fire alarms, and gunshot sounds. Primary data were obtained through direct audio recording to ensure consistency with real-world environmental conditions. In addition, publicly available datasets, such as UrbanSound8K and ESC-50, were utilized as secondary data sources to increase data diversity and enhance the performance of the developed model.

3.3. Data Preprocessing

The data preprocessing stage was carried out to ensure that the collected audio data are of high quality and formatted appropriately for model training. This stage consists of two main processes: data cleaning and feature extraction. Data cleaning aims to remove irrelevant or low-quality audio samples. Subsequently, feature extraction is performed using the Short-Time Fourier Transform (STFT), which converts audio signals into spectrogram representations. These spectrograms serve as input to the Convolutional Neural Network (CNN) model for detecting frequency patterns in the time–frequency domain.

3.4. Model Training

The CNN model training stage aims to develop a machine learning architecture capable of recognizing and classifying emergency sounds. This process begins with defining the model architecture, including the number of convolutional layers, pooling layers, and fully connected layers. The convolutional layers are responsible for extracting essential features from the spectrograms, while the pooling layers reduce data dimensionality to improve computational efficiency without significant loss of important information.

3.5. Model Evaluation

The model evaluation stage aims to assess the performance of the Convolutional Neural Network (CNN) in detecting and classifying emergency sounds before deployment in the actual system. This process is conducted by testing the model using validation data, which consist of data not used during training to measure the model's ability to generalize to unseen data. Model performance is evaluated based on training accuracy and validation accuracy.

3.6. Android Application Development

The Android application development stage aims to design and implement an application capable of detecting emergency sounds. This process includes designing the user interface (UI), integrating the trained Convolutional Neural Network (CNN) model converted into TensorFlow Lite format (.tflite), and testing the application's performance on Android devices.

3.7. System Testing

System testing is conducted to evaluate the performance and accuracy of the Android application in detecting emergency sounds and to ensure that all features function correctly. The method used in this study is Black Box Testing, which focuses on testing system functionality without examining the source code or internal structure.

4. Results and Discussion

4.1. Data Collection

The initial phase of this study began with the collection of a dataset used as the basis for training and testing the CNN model. The dataset consists of two types of data sources: primary data and secondary data. The following is the distribution of data for each type of emergency call:

Table 1: Dataset Composition by Class Type

Class	Amount of Data
Siren	100
Fire Alarm	100
Horn	100
Gun Shot	100
Noise	100
Total	**500**

The initial dataset used in this study consists of 500 audio samples divided into five categories: siren, fire alarm, horn, gunshot, and noise. The noise category includes various types of non-alarm sounds, such as human conversation, environmental noise, vehicle sounds, whistles, and other sounds commonly heard in the surrounding environment.

4.2. Data Preprocessing

a. Splitting Dataset

The first step in the data preprocessing phase is to split the data into training data and testing data. The test was conducted by comparing two data split scenarios: an 80:20 ratio and a 90:10 ratio. With the 80:20 ratio, the dataset was split into 400 data points for training and 100 data points for testing, while with the 90:10 ratio, the dataset was split into 450 data points for training and 50 data points for testing.

b. Data Augmentation

Data augmentation was applied exclusively to the training set to increase sample diversity and reduce the risk of overfitting, utilizing the audiomentations library. Five transformation techniques were applied with a probability of 0.5 per technique, generating three augmented variants per original sample. The techniques employed were: (1) Gaussian Noise (amplitude range: 0.001–0.015) to simulate noisy acoustic environments; (2) Time Stretching (80%–120%) to represent tempo variations in real-world conditions; (3) Pitch Shifting (−2 to +2 semitones) to account for differences across recording devices; (4) Time Shifting (±30% of duration) to simulate variability in sound detection onset; and (5) Gain Adjustment (±6 dB) to represent intensity variations across different recording distances.

c. Spectrogram Conversion

Prior to model input, all audio signals were converted into spectrograms through a four-stage process: (1) signal preprocessing, including trimming and resampling to 16 kHz; (2) application of the Short-Time Fourier Transform (STFT) with a frame length of 255 and a frame step of 128 to capture time-varying frequency components; (3) magnitude extraction to represent signal energy in the frequency domain; and (4) channel dimension expansion to reshape the data from two dimensions to three dimensions in the format (*height*, *width*, *channels*), as required by the CNN input layer. The resulting spectrograms provide a two-dimensional visual representation of frequency energy distribution, where color intensity encodes the energy level at each time-frequency coordinate.

4.3. Model Training

The CNN architecture was constructed as follows. An initial Resizing layer was applied to standardize the spatial dimensions of spectrogram inputs, followed by a Normalization layer to scale pixel values based on the mean and standard deviation of the training data. Three successive convolutional layers with filter sizes of 16, 32, and 64, respectively — each employing a 3×3 kernel and ReLU activation were used for hierarchical feature extraction. Each convolutional layer was followed by a MaxPooling2D layer to reduce feature map dimensionality, and a Dropout layer with a rate of 0.25 to mitigate overfitting. The extracted feature maps were subsequently flattened into a one-dimensional vector and passed through a fully connected Dense layer comprising 128 neurons with ReLU activation, followed by an additional Dropout layer with a rate of 0.5 to further strengthen regularization. The final output layer employed a Softmax activation function, with the number of neurons corresponding to the total number of target classes.

Table 2: CNN Model Hyperparameter Configuration

Parameter	Value
Batch Size	32
Learning Rate	0.0001
Epochs	100
Optimizer	Adam
Loss Function	Sparse Categorical Cross-Entropy

Training was conducted under two dataset splitting scenarios. The 80:20 split yielded the highest accuracy of 97.22% with a loss value of 0.1016, while the 90:10 split produced an accuracy of 96.57% with a loss of 0.1203. Analysis of the training accuracy and loss curves indicated that the model generalized effectively without evidence of overfitting or underfitting, as reflected by the relatively narrow gap between training accuracy (97.22%) and validation accuracy (91%), alongside a consistently declining loss trajectory throughout the training process.

4.4. Model Evaluation

Model evaluation was conducted to assess the generalization capability of the CNN on data that had not been seen during training. Performance was measured using four metrics — precision, recall, F1-score, and overall accuracy — complemented by a confusion matrix to visualize the distribution of prediction results across all sound classes.

Analysis of the confusion matrix revealed the following per-class findings. The gunshot class achieved perfect classification, with all 35 test samples correctly identified (recall = 1.00), attributed to its highly distinctive impulsive waveform characterized by an abrupt amplitude spike and extremely short duration. The fire alarm class also demonstrated strong performance, with 48 out of 49 samples correctly classified (precision = 0.94, recall = 0.98, F1-score = 0.96), with only one sample misclassified as a car horn. The noise class achieved a perfect recall of 1.00, indicating that all non-emergency sounds were correctly identified, although a slightly lower precision (0.88) suggests that a small number of samples from other classes were incorrectly predicted as noise. The siren class yielded a precision of 0.94 and recall of 0.84 (F1-score = 0.89), with some samples misclassified as car horn or noise, likely due to the shared periodic frequency characteristics between these classes. The car horn class exhibited the lowest performance, with 33 out of 41 samples correctly classified (precision = 0.92, recall = 0.80, F1-score = 0.86), reflecting the spectral similarity between car horn and other classes such as sirens and fire alarms.

Overall accuracy was computed as follows:

$$Accuracy = \frac{193}{208} \times 100\% \approx 93\%$$

Table 3: CNN Model Evaluation Metrics per Class

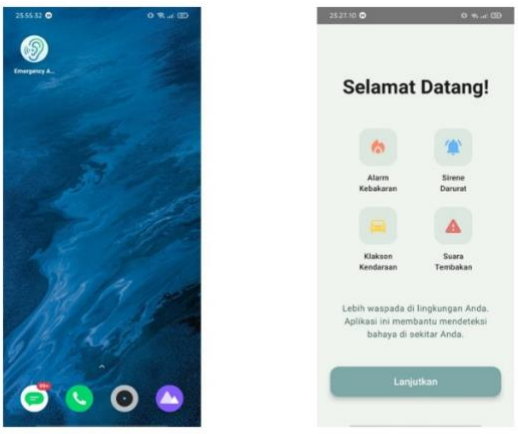
Class	Metrics			
	Precision	**Recall**	**F1-Score**	**Accuracy**
Siren	0.94	0.84	0.89	
Fire Alarm	0.94	0.98	0.96	
Horn	0.92	0.80	0.86	**0.93**
Gun Shot	0.97	1.00	0.99	
Noise	0.88	1.00	0.94	

4.5. Android Application Development

The system was implemented to ensure that the emergency sound detection app could be used by people who are deaf by presenting sound detection results in a clear visual format. The model, which had been previously trained and evaluated, was then converted to the TensorFlow Lite format so it could be implemented on Android devices. The conversion process was performed from the HDF5 (.h5) file format to the TensorFlow Lite (.tflite) format using the TFLiteConverter tool from the TensorFlow Lite Python API. The conversion results in a model file of 5.5 MB, making it lightweight enough for implementation on Android devices. The converted model is then implemented on Android devices to perform real-time inference in classifying emergency sounds for the deaf.

1. Main Page

To open this sound detection app, tap the app's logo on your Android device. Once opened, the main screen will appear as shown in Figure 3. On this main screen, you'll see the message "Selamat Datang" and a list of emergency sounds that can be detected. There is also a "Lanjutkan" button to proceed to the detection screen.



(a) (b)

Fig. 3: (a) App Icon and (b) Main Page

2. Detection Page (Before Detection Begins)

The detection interface presents a simple and user-friendly layout prior to initiating the detection process. A title, “Emergency Sound Detection,” is displayed at the top to provide context. The main component is a centrally positioned microphone icon, which serves as the trigger for audio recording. Additionally, the text “Press the button to start” guides users on how to initiate the detection process.

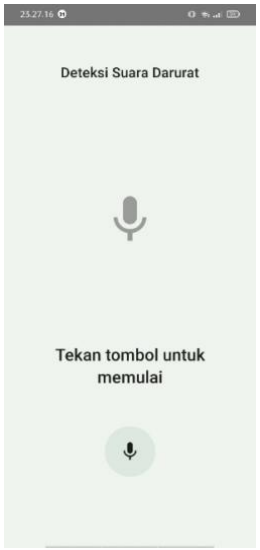


Fig. 4: Detection Page

3. During the Detection Process

After the recording button is pressed, the interface transitions to the detection mode, indicated by the status text “Listening…,” which signifies that the application is capturing real-time audio for processing by the CNN model. A subtle circular animation is displayed at the center of the screen to indicate active system monitoring. The classification result is presented at the bottom of the screen; for instance, “No emergency sound detected” indicates the absence of emergency audio. The microphone button remains available to stop or restart the detection process, ensuring intuitive user control.

Fig. 5: Display during the Detection Process

4. Display When an Emergency Sound Is Detected

Figure 6 illustrates the condition in which the application successfully detects an emergency sound. The system is designed to deliver clear and easily interpretable visual alerts, particularly for individuals with hearing impairment. Upon detection, an Android system notification displaying “WARNING!” appears in the notification bar, accompanied by device vibration to ensure immediate user awareness. Simultaneously, a pop-up alert occupies the main interface, triggered automatically once the CNN model classifies the detected sound as hazardous.

Fig. 6: Detected Emergency Sound

4.6. System Testing

System testing is conducted to ensure that the developed system operates in accordance with the defined functional requirements. The system testing method used is Black-Box Testing, which focuses on inputs and outputs without considering the system's internal processes.

Table 4: System Testing Results

No	Test Scenario	User Action	Expected Result	Status
1	Testing whether the application can initiate the listening process	The user presses the microphone button on the main page	The application changes to "Listening..." status	Successful
2	Testing the application's ability to recognize siren sounds	Play a siren sound	The application displays "Siren" detection result, along with a warning pop-up and notification	Successful
3	Testing the application's ability to recognize fire alarm sounds	Play a fire alarm sound	The application displays "Fire Alarm" detection result, along with a warning pop-up and notification	Successful
4	Testing the application's ability to recognize horn sounds	Play a horn sound	The application displays "Horn" detection result, along with a warning pop-up and notification	Successful
5	Testing the application's ability to recognize gunshot sounds	Play a gunshot sound	The application displays "Gunshot" detection result, along with a warning pop-up and notification	Successful
6	Testing whether background noise is not classified as emergency sound	Play non-emergency sounds (crowd noise, wind, vehicles)	The application displays "No emergency sound detected"	Successful
7	Testing whether a notification appears when an emergency sound is detected	Play any emergency sound	Android notification appears: "Warning!"	Successful
8	Testing system behavior after closing the alert pop-up	The user presses "I Understand" button	The pop-up closes and the application resumes listening	Successful
9	Testing whether low-volume emergency sounds can be detected	Play an emergency sound at low volume	The application displays the correct detection result	Successful
10	Testing whether the application remains stable when receiving multiple sounds	Play different sounds sequentially	The application continues running without errors	Successful
11	Testing audio access permission	Run the application for the first time	The application requests microphone permission	Successful
12	Testing application behavior when microphone access is denied	Press "Deny" when permission is requested	The application displays a message indicating that permission is required	Successful

System testing in a real-world environment was also conducted to evaluate the performance of the emergency sound detection application when run directly on Android devices. The purpose of this testing was to assess the model's consistency under actual usage conditions, taking into account variations in devices, environments, and the distance between the sound source and the device. This testing was conducted at Al-Hadi Homeschooling.

Fig. 7: System Testing at Al-Hadi Homeschooling

5. Conclusion

Based on the conducted study, it can be concluded that the proposed system successfully detects emergency sounds using a Convolutional Neural Network (CNN). The model is capable of recognizing four types of emergency sounds sirens, fire alarms, vehicle horns, and gunshots—achieving a best accuracy of 97.17% and an average accuracy of 93%. The CNN demonstrated stable performance and effectively captured distinctive sound patterns from spectrogram representations while avoiding overfitting and underfitting.

Furthermore, the trained CNN model was successfully deployed in an Android application using TensorFlow Lite, enabling real-time inference on mobile devices. The application integrates audio acquisition, data processing, and model inference to provide visual notifications, alert pop-ups, and vibration when emergency sounds are detected. This implementation demonstrates the system's potential as an assistive tool for individuals with hearing impairment in recognizing emergency situations.

References

[1] A. Shukla *et al.*, "Hearing Loss, Loneliness, and Social Isolation: A Systematic Review," May 01, 2020, *SAGE Publications Inc.* doi: 10.1177/0194599820910377.

[2] World Health Organization, "WORLD REPORT ON HEARING," 2021. [Online]. Available: https://youtu.be/EmXwAnP9puQ

[3] H. B. M. Mohammed and N. Cavus, "Utilization of Detection of Non-Speech Sound for Sustainable Quality of Life for Deaf and Hearing-Impaired People: A Systematic Literature Review," Oct. 01, 2024, *Multidisciplinary Digital Publishing Institute (MDPI)*. doi: 10.3390/su16208976.

[4] J. Wang and G. Goldsztein, "Audio Classification of Bird Species Using Convolutional Neural Networks," *Journal of Student Research*, vol. 12, no. 1, 2023, [Online]. Available: www.JSR.org

[5] Z. Liu, "Audio Feature Extraction and Classification Technology Based on Convolutional Neural Network," *Journal of Electrical Systems*, pp. 1425–1431, 2024.

[6] R. H. Rafliansyah, B. Rahmat, and C. A. Putra, "Klasifikasi Suara Instrumen Musik Tiup Menggunakan Metode Convolutional Neural Network," *Merkurius : Jurnal Riset Sistem Informasi dan Teknik Informatika*, vol. 2, no. 4, pp. 01–09, Jun. 2024, doi: 10.61132/merkurius.v2i4.119.

[7] A. Sehgal and N. Kehtarnavaz, "A Convolutional Neural Network Smartphone App for Real-Time Voice Activity Detection," *IEEE Access*, vol. 6, pp. 9017–9026, Jan. 2018, doi: 10.1109/ACCESS.2018.2800728.

[8] Suharsiwi, *PENDIDIKAN ANAK BERKEBUTUHAN KHUSUS*. Yogyakarta: CV Prima Print, 2017.

[9] S. Soemantri, *Psikologi Anak Luar Biasa*. Bandung: Reflika Aditama, 2007.

[10] Badan Standardisasi Nasional (BSN), "SNI 03-6574-2001: Tata Cara Perancangan Pencahayaan Darurat, Tanda Arah dan Sistem Peringatan Bahaya pada Bangunan Gedung," Badan Standardisasi Nasional, 2001.

[11] G. Ballou, *Handbook for Sound Engineers Fourth Edition*. Routledge, 2008.

[12] International Organization for Standardization (ISO), "ISO 7731:2003 - Ergonomics Danger signals for public and work areas — Auditory danger signals," 2003.

[13] M. Aykanat, Ö. Kılıç, B. Kurt, and S. Saryal, "Classification of lung sounds using convolutional neural networks," *EURASIP J. Image Video Process.*, vol. 2017, no. 1, Dec. 2017, doi: 10.1186/s13640-017-0213-2.

[14] B. Boashash, *Time-Frequency Signal Analysis and Processing: A Comprehensive Reference*. Academic press, 2015.

[15] I. Goodfellow, Y. Bengio, A. Courville, and Y. Bengio, *Deep learning*. Cambridge: MIT Press, 2016.

[16] J. DiMarzio, *Beginning Android Programming with Android Studio*. John Wiley & Sons, 2017. [Online]. Available: www.wowebook.orgWOW!eBookwww.wowebook.org

[17] Android Developers, "Dasar-dasar aplikasi," Android. Accessed: Feb. 24, 2025. [Online]. Available: https://developer.android.com/guide/components/fundamentals?hl=id