UNIVERSITY of HOUSTON

COLLEGE of NATURAL SCIENCES & MATHEMATICS

COSC 6364

ADV. NUMERICAL ANALYSIS PROJECT

FINAL REPORT

TEAM MEMBERS:

SHEEMA ANUSH (2303265)

FARAAZ REHAN JUNAIDI MOHAMMED (2297877)

1. ABSTRACT

Emergency audio recognition systems play a crucial role in promptly identifying emergency sounds and notifying first responders for swift action, thus minimizing emergency response time. However, existing systems encounter various challenges including high false positive rates, low accuracy, and difficulty in distinguishing emergency sounds amidst noisy environments. To overcome these hurdles, this study aims to assess the efficacy of an emergency audio recognition system employing frequency-based signal processing techniques. The Fast Fourier Transform (FFT) algorithm is utilized to convert discrete time-domain signals into the frequency domain, enabling the modeling of various parameters for analytical feature extraction. The primary objective is to analyze the system's performance in noisy conditions by preprocessing an audio dataset with varying levels of random noise, thereby generating diverse datasets for model training and evaluation. This investigation seeks to ascertain the viability of frequency-based signal processing in developing a robust and precise emergency audio recognition system for real-time emergency detection and response.

2. INTRODUCTION

Audio data holds significant importance as it offers valuable insights into the acoustic environment where it was captured. These insights serve various applications like speech recognition, audio signal processing, music analysis, and emergency audio recognition.

In the domain of emergency audio recognition, audio data plays a pivotal role in swiftly detecting and responding to urgent situations. These systems analyze audio data in real-time to identify emergency sounds such as ambulance sirens, fire truck horns, gunshots, explosions, and other relevant sounds.

Frequency-based signal processing stands out as a key technique in emergency audio recognition systems. This method involves dissecting audio signals into their frequency components, facilitating the recognition of distinct patterns and characteristics associated with different types of sounds.

This approach offers several advantages, including real-time processing capability, high efficiency, and accurate sound detection and classification. These qualities are crucial for ensuring prompt and precise alerts to emergency incidents.

The adoption of convolutional neural networks (CNNs) for audio signal processing has garnered attention due to their proficiency in handling spectrogram images of audio signals. Spectrogram images depict the frequency distribution of audio signals over time, serving as input data for CNN models in emergency audio signal detection.

CNN models excel in learning intricate patterns and features within spectrogram images, enabling them to make precise predictions based on learned characteristics. Their scalability and capacity to process large datasets make them well-suited for handling the vast volume of audio data in emergency scenarios.

By automating the emergency response process, these systems diminish the reliance on human intervention and enhance the efficiency of emergency services, particularly in resource-constrained environments or situations where rapid response is imperative.

They contribute to public safety by functioning as early warning systems for events like gun violence or explosions. Leveraging spectrogram images with CNN models in emergency audio recognition represents a pivotal application of audio signal processing, promising substantial improvements in emergency response efficiency, public safety, and lives saved.

3. METHODOLOGY

Our methodology encompasses the following stages:

1. Acquisition and preprocessing of the raw dataset referenced in [1].
2. Conversion of the raw audio dataset into spectrogram images in JPG format.
3. Visualization of the audio data through Fast Fourier Transform (FFT) and convolution with noise and cross signals.
4. Development of a custom Convolutional Neural Network (CNN) utilizing a combination of Conv2D layers, Maxpooling layers, and Dense layers.
5. Training of the custom model on the labeled training data categorized into "emergency" and "non-emergency."
6. Assessment of the model's performance in detecting and classifying audio signals as either emergency or non-emergency.
7. Introduction of noise at varying levels (e.g., 15%, 30%, 45%, and 60%) and evaluation of the model's robustness against noisy data.
8. Comparison of results obtained from the custom-built CNN model with those from pre-trained state-of-the-art CNN architectures such as ResNet50 and VGG16.

A. Data Collection and Preparation

The initial dataset comprising audio recordings of emergency vehicle sounds, each clip spanning 3 seconds in duration, was obtained from [1]. Figure 1.1 illustrates the dataset utilized in our emergency audio detection system.

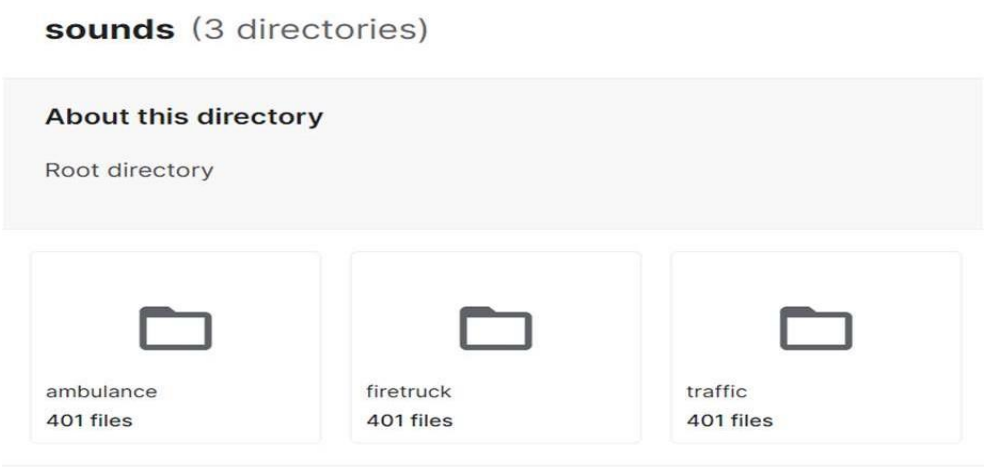

Figure 1.1: Raw data available on Kaggle.

The original dataset from [1] lacked organization and class categorization. To meet the prerequisites for CNN model input, we restructured the dataset accordingly. The dataset is now divided into two classes: emergency and non-emergency, with each class stored in separate directories. Each directory contains 200 audio files corresponding to the respective class. Figure 1.2 presents the organized data utilized by the CNN models.

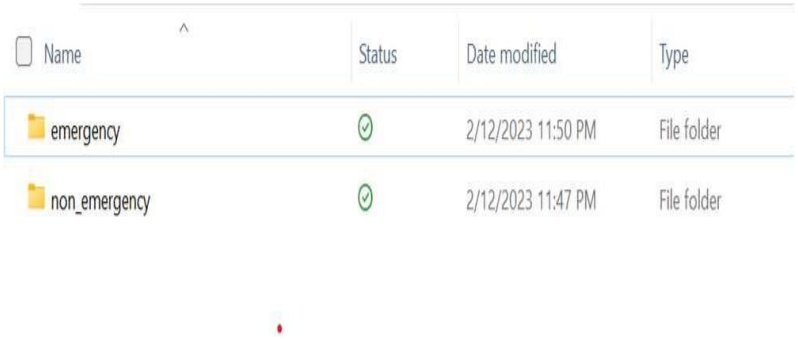

Figure 1.2: Audio dataset.

Spectrogram images visually represent the frequency content of an audio signal over time, serving as vital input for CNN models in detecting emergency audio signals [6]. Following the sorting of audio .wav files into their respective directories, we executed a script to generate spectrograms for each audio file. Figure 2 showcases samples of the produced spectrograms.

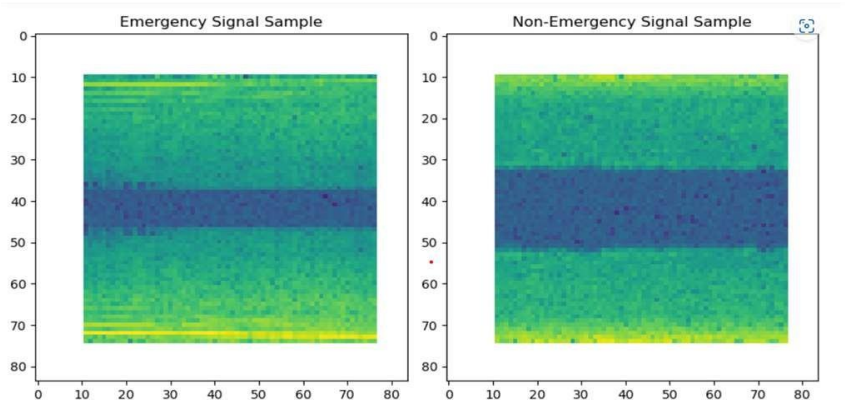

Figure 2: Spectrogram Samples.

B. Data Exploration and Visualization

In the initial phase of implementation, we acquired the raw data from [1] in the form of JPEG files. The raw data comprised over 600 audio files containing both emergency and non-emergency datasets in .wav format. Subsequently, we categorized the data according to the requirements of our CNN input and generated spectrograms to serve as inputs to our network.

The sound waves emitted by emergency vehicles like firetrucks and ambulances exhibit similar spectral characteristics, capturing comparable frequencies. This inherent similarity enables their distinction from other sounds within the acoustic environment. In the realm of audio signal processing, this shared spectral property can be harnessed to develop an emergency audio recognition system using cross-correlation. Figure 3 illustrates the Fast Fourier Transform (FFT) of the audio signals.

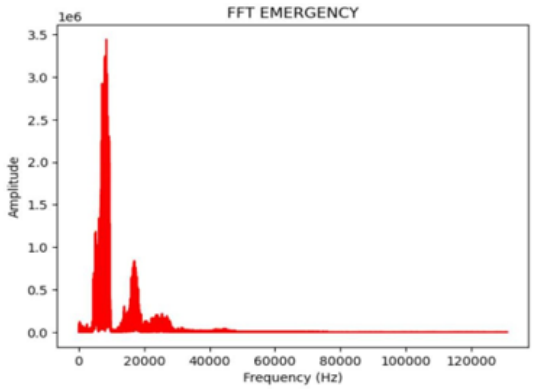

Figure 3.1: FFT of Emergency Audio Signal.

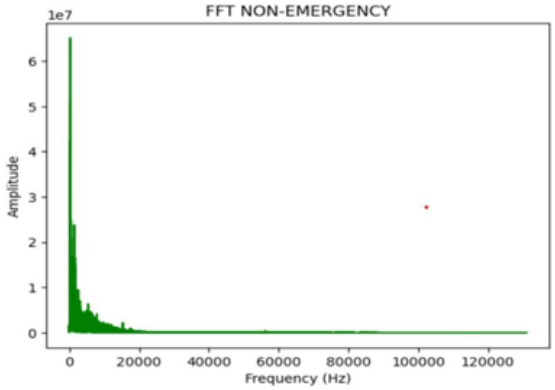

Figure 3.1: FFT of Non-Emergency Audio Signal.

We conducted an experiment to assess the resemblance between emergency and non-emergency audio signals, both with and without noise. Cross-correlation entails convolving the Fast Fourier Transform (FFT) [7] of a sound wave with a corresponding FFT of itself, aligning the two signals at positions corresponding to vectors between peaks of density. During this process, the peaks of density align and reinforce each other, culminating in a singular peak in the correlation function.

Our findings indicated that when the emergency signal with noise exhibited greater similarity to the emergency signal itself compared to the non-emergency signal, the cross-correlation yielded a unimodal peak. This observation suggests that both signals peaked at the same frequency level, indicating their similarity. Figure 4 illustrates this phenomenon.

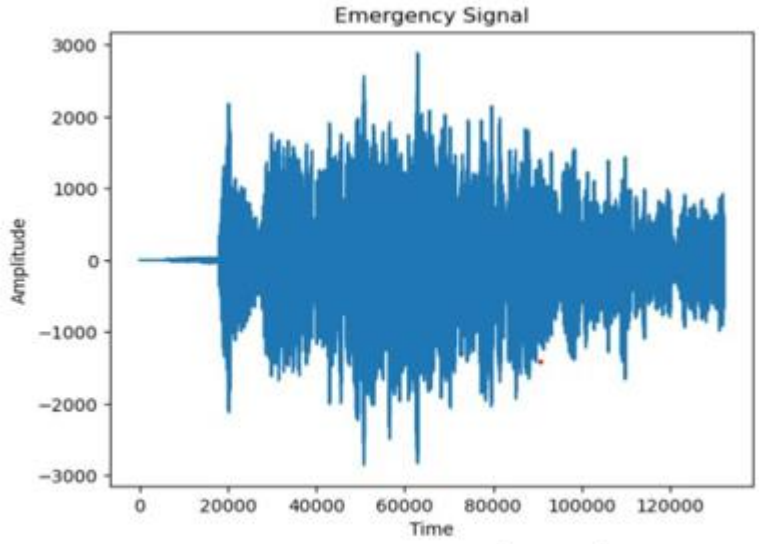

Figure 4.1: Emergency Audio Signal.

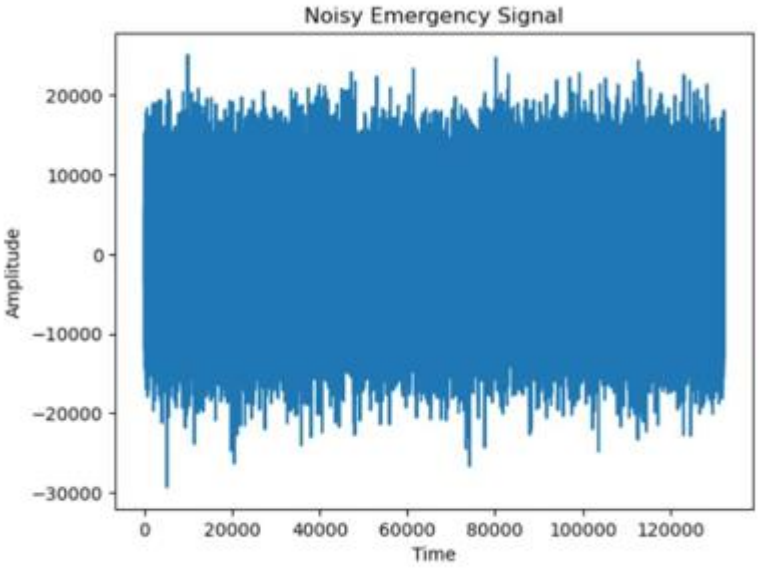

Figure 4.2: Emergency Audio Signal convolved with Noise.

Conversely, when we convolve the FFT of a dissimilar signal with the original signal, the outcome disperses into multiple modes instead of forming a single peak in the correlation function. This divergence occurs due to the dissimilar spectral composition of the signal, where the peaks of density fail to align with each other.

In our experiment, when the emergency signal with noise exhibited greater similarity to no signal rather than the non-emergency signal, the cross-correlation resulted in a bimodal peak. This outcome suggests that both signals peaked at different frequency levels, indicating their dissimilarity in a relative sense. Figure 5 illustrates this phenomenon.

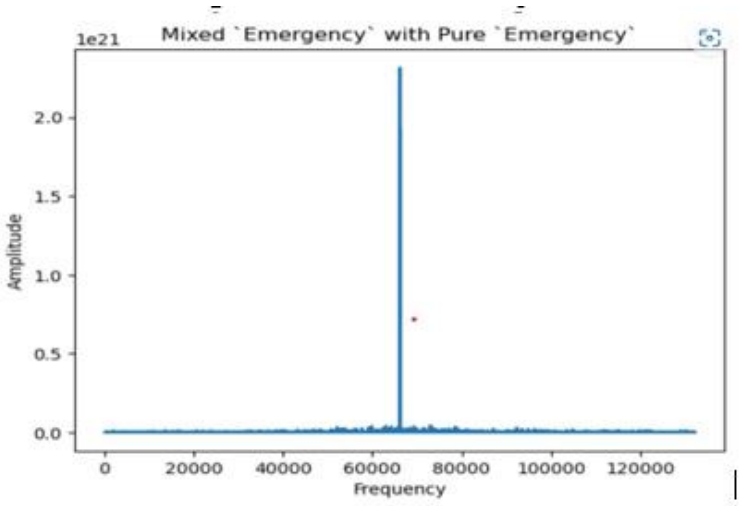

Figure 5.1: Convolution of Emergency Signal with Noisy Emergency Signal.

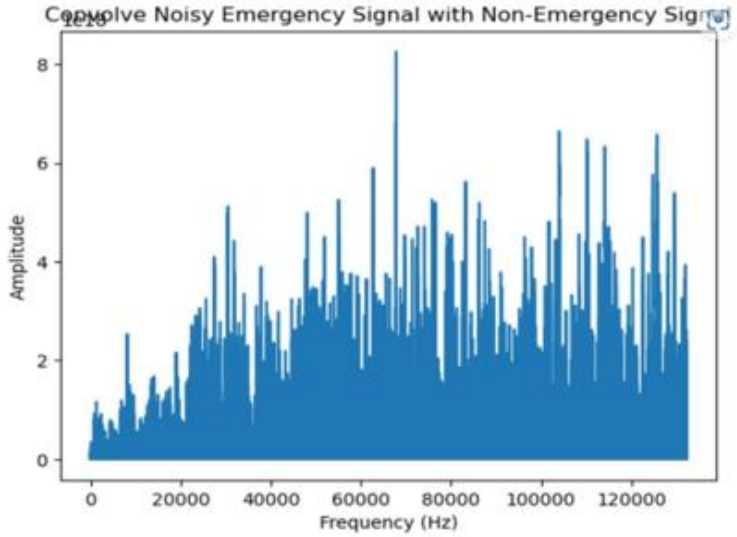

Figure 5.2: Convolution of Noisy Emergency Signal with Non-Emergency Signal.

These results suggest that a simple comparison of the peaks of convoluted signals can help us label a data point as either an emergency or non-emergency signal. This has important implications for the development of emergency audio recognition systems, as it provides a straightforward method for distinguishing between emergency and non-emergency sounds.

C. Emergency Audio Recognition System

Convolutional Neural Networks (CNNs) represent an advanced form of deep learning algorithms that leverage convolution principles. These algorithms are tailored to analyze and categorize visual or auditory data by discerning significant features within the input image or signal. This process involves the allocation of adjustable weights and biases to

different elements or aspects within the image, empowering the network to effectively distinguish between various object categories.

In the realm of audio recognition, the transformation of a signal into a spectrogram provides a graphical representation of its frequency content over time [8]. This facilitates the network's analysis of the audio signal's characteristics in the frequency domain, which is instrumental for distinguishing between different types of sounds. Spectrograms are constructed by applying Fast Fourier Transforms (FFTs) to the audio signal at various time intervals, as demonstrated in Figs 3, 4, and 5, and subsequently overlapping them to generate a two-dimensional image. The horizontal axis represents time, the vertical axis depicts frequency, and the intensity of each pixel reflects the signal's amplitude at that specific point.

For the binary classification of audio signals, a CNN [9] can be specifically designed to accept spectrograms as input and classify them as either 'emergency' or 'non-emergency' in our scenario. This type of network undergoes training on a dataset comprising labeled audio samples, with each sample being categorized into one of the two classes, as illustrated in Figure 6. Throughout the training process, the network iteratively adjusts its learnable parameters to minimize the disparity between its predictions and the actual labels of the training data.

Out[18]:

	filename	category
0	emergency_1.jpg	emergency
1	emergency_10.jpg	emergency
2	emergency_100.jpg	emergency
3	emergency_101.jpg	emergency
4	emergency_102.jpg	emergency
...	...	...
390	non_emergency_90.jpg	non_emergency
391	non_emergency_91.jpg	non_emergency
392	non_emergency_92.jpg	non_emergency
393	non_emergency_93.jpg	non_emergency
394	non_emergency_94.jpg	non_emergency

395 rows × 2 columns

Figure 6: Categorized labeled data.

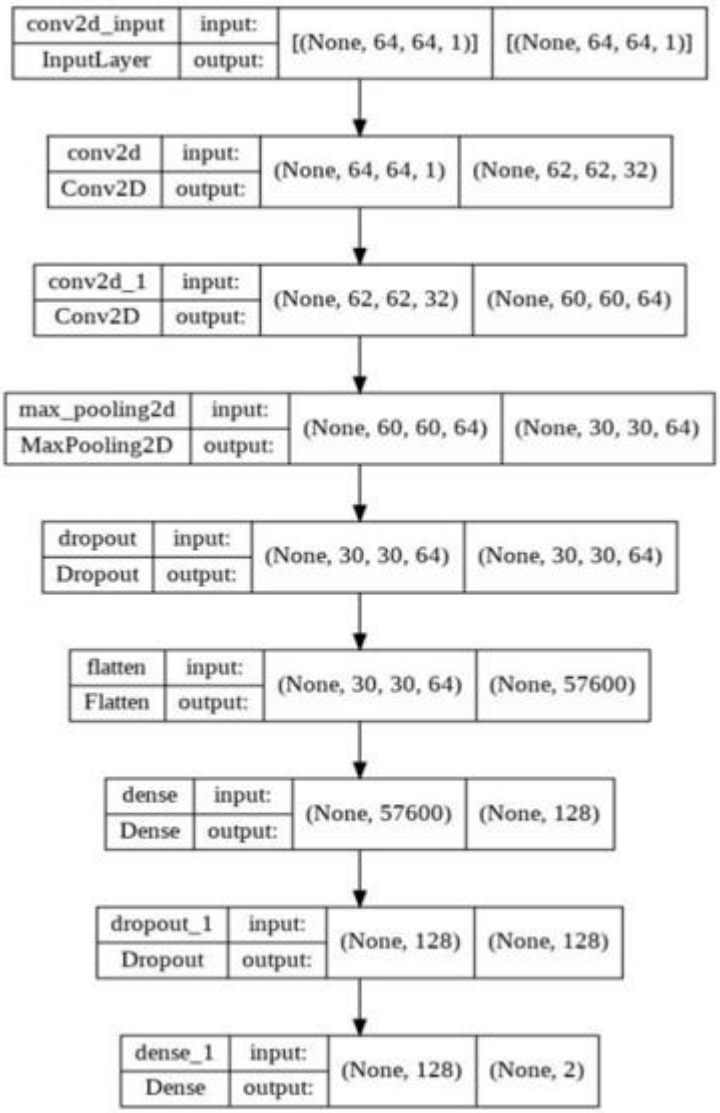

Figure 7: Custom-built CNN architecture.

The implemented CNN model incorporates several key features:

- **Max-pooling operation:** This operation aids in summarizing features and reducing computations by selecting the most prominent features from the feature maps. It involves sliding a filter across the feature map and selecting the maximum value in each window. By retaining only the most significant features, this operation helps to decrease the computational cost of the network.

- **Dropout:** Dropout is a regularization technique that mitigates overfitting by randomly deactivating a fraction of neurons at each training step. This prevents the network from overly relying on individual neurons or features, thus enhancing the model's generalization capability. In our model, dropout rates of 0.25 and 0.50 were utilized.

- **Flatten operation:** This operation transforms the 2D feature map into a 1D vector suitable for input into the fully connected (FC) layer. The FC layer consists of neurons that receive input from the preceding layers' neurons. Flattening the feature map ensures compatibility with the input of the FC layer, with neurons in the FC layer containing values representing the probability of specific features being present in the image.

- **Dense layer:** The dense layer comprises neurons that receive input from the preceding layers' neurons. Responsible for classification, the dense layers leverage extracted features from convolution. The first dense layer, Dense(128), assesses feature relevance, while the last layer, Dense(2), determines probability scores for membership in specific classes, such as 'emergency' and 'non-emergency'. Activation functions employed in our model include ReLu and SoftMax, where ReLu reduces computational cost in CNNs and SoftMax facilitates multi-class classification by providing probabilities for each class.

- **Parameters:** Parameters encompass weights and biases utilized in training the CNN model, crucial for learning and making accurate predictions. During training, the CNN updates these parameters to minimize error between predicted and actual outputs. Optimized values of parameters are determined during training and significantly influence model accuracy.

D. Pre-trained Models

VGG16: The VGG16 model represents a convolutional neural network (CNN) renowned for its effectiveness in image classification tasks, notably demonstrated on the ImageNet dataset encompassing over 14 million images across 1000 classes. Leveraging transfer learning with the pre-trained VGG16 model [14], depicted in Figure 8, we apply it to our spectrogram dataset following the methodology outlined in [2]. Transfer learning allows us to capitalize on the learned weights and features of the pre-trained VGG model to classify our spectrograms into "emergency" and "non-emergency" categories. To retain the knowledge embedded in the pre-trained model, we freeze its layers and introduce new trainable layers atop to translate existing features into predictions for our dataset. Subsequently, we unfreeze select layers of the VGG model and jointly train them with the new layers to refine the higher-order feature representations in the base model, rendering them more pertinent to our specific task.

ResNet50: Alongside the VGG16 model, we incorporate the ResNet50 model [15] as another pre-trained model for the same classification objective. ResNet50, depicted in Figure 9, is also a CNN-based model recognized for its efficacy in image classification tasks. Employing a similar transfer learning approach, we initialize the ResNet50 model with its pre-trained weights, freeze its layers, append new trainable layers, and refine the model's higher-order features through joint training.

By harnessing pre-trained models such as VGG16 and ResNet50, we can expedite and enhance classification performance, requiring less training data and time while achieving superior accuracy.

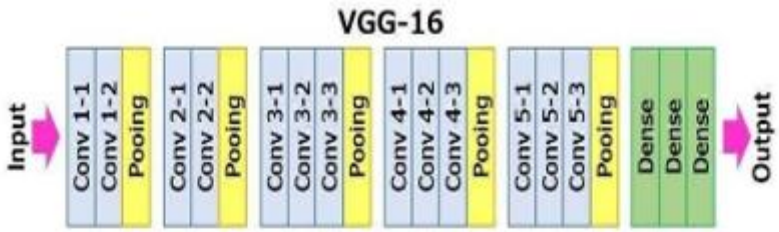

Figure 8: VGG16 Architecture.

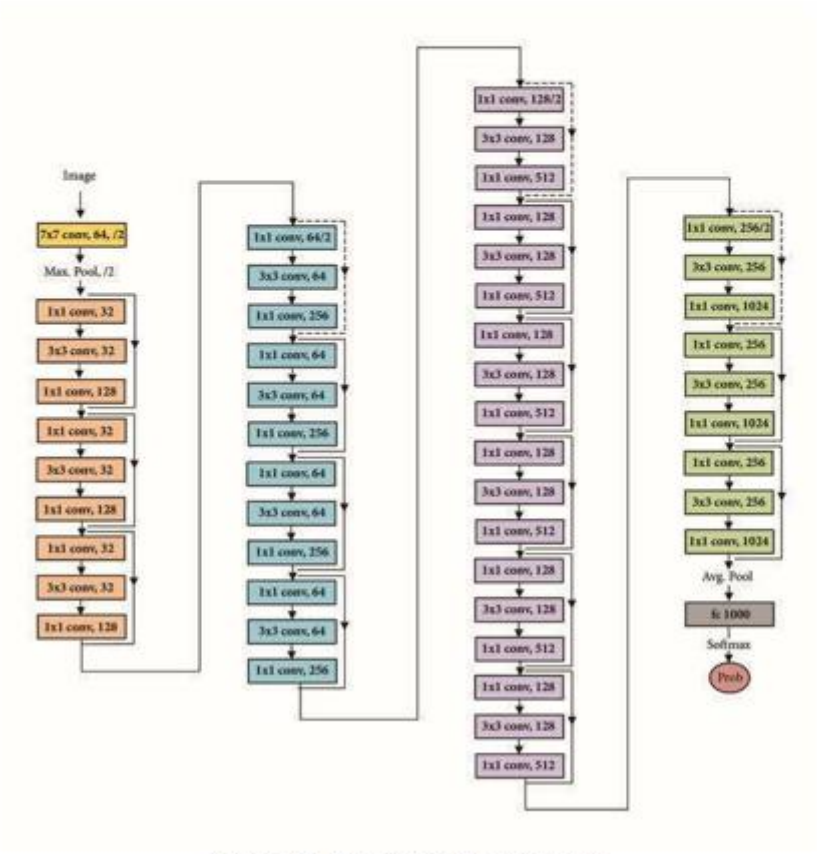

Figure 9: ResNet50 Architecture.

4. RESULTS

A. Custom-built CNN Model

Accuracy: Upon training the custom-built Convolutional Neural Network (CNN) model for 15 epochs, the training accuracy attains 100%, indicating proficient learning on the training data. Furthermore, the test accuracy [16] achieves a commendable 98.75%, signifying robust generalization to new, unseen data.

It's noteworthy that the reported accuracy outcomes are derived from employing a train-test split strategy, where 64% of the data was allocated for training, 20% for

testing, and 16% for validation. The dataset partitioning was conducted using the train_test_split() function, which randomizes the dataset and allocates samples to the training, testing, and validation sets according to the specified ratios.

Confusion Matrix: The classification model's performance distribution can be succinctly summarized using a confusion matrix [17], offering vital metrics such as True Positive, True Negative, False Positive, and False Negative values. In our investigation, a confusion matrix was generated, and the results are depicted in Figure 10.

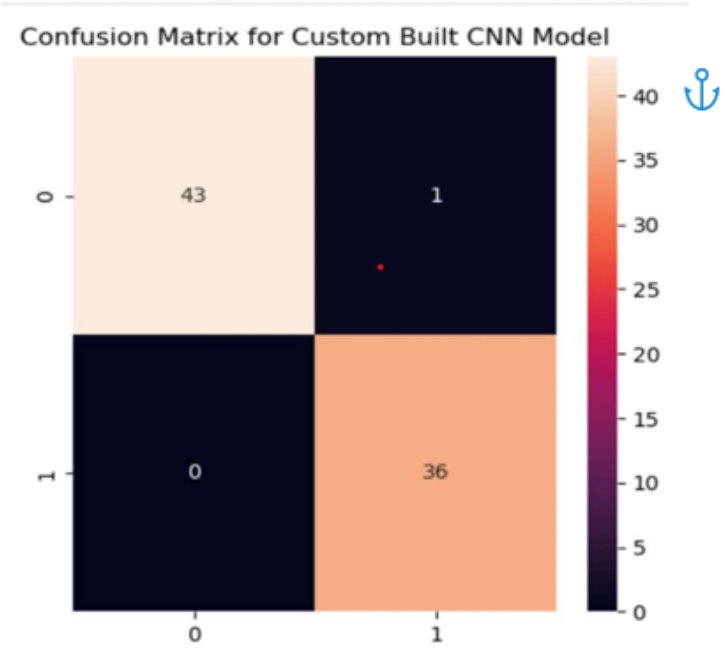

Figure 10: Confusion Matrix for Custom-built CNN Model.

From the confusion matrix provided in Figure 11, we can compute significant evaluation metrics such as precision, recall, and F1 score.

Precision quantifies the proportion of correctly classified positive samples out of all the classified positives. In our scenario, the precision score for our model is 0.97, implying that our model accurately classified nearly all the positive samples, with only one misclassification.

Recall evaluates the number of actual positive samples correctly identified as positive. In our case, the recall score for our model is 1, indicating that all actual positive samples were correctly classified.

```
precision = precision_score(y_test, prediction)
print('Precision: %f' % precision)
# recall: tp / (tp + fn)
recall = recall_score(y_test, prediction)
print('Recall: %f' % recall)
# f1: 2 tp / (2 tp + fp + fn)
f1 = f1_score(y_test, prediction)
print('F1 score: %f' % f1)
```

```
Precision: 0.972973
Recall: 1.000000
F1 score: 0.986301
```

Figure 11: Precision, Recall and F1 Score.

F1 Score serves as a metric that considers both precision and recall, offering a balanced assessment. It proves particularly valuable when dealing with unbalanced class distributions. In our context, the F1 score calculates to be 0.98, indicating a high balance between the model's predicted classifications.

Performance under induced noise levels: The model has undergone comprehensive testing to ascertain its accuracy in identifying signals under various conditions.

To further assess the model's performance, we subjected it to rigorous testing by inducing noise at different levels: 5%, 15%, 25%, and 35%, respectively. Subsequently, we plotted its accuracy against varying noise levels. This analysis furnishes crucial insights into the model's efficacy in real-world scenarios where audio signals may encounter distortion due to background noise. The plot presented in Figure 12 offers a visual depiction of how the model's accuracy fluctuates with increasing noise levels.

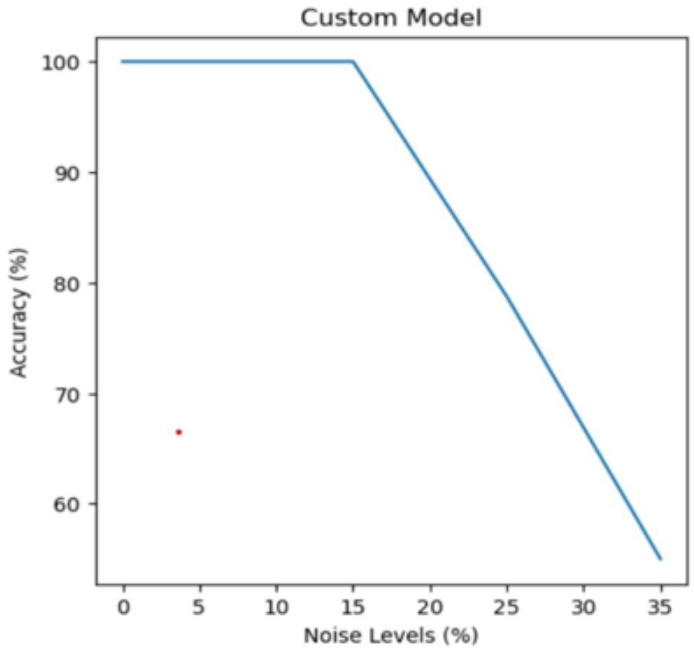

Figure 12: Accuracy v/s Noise Levels for Custom Model.

Overall, our binary classification custom model has proven to be effective in accurately classifying emergency and non- emergency audio signals and provides a promising solution for real-world applications where quick and accurate identification of such sounds is crucial.

B. Pre-trained ResNet50

Accuracy: Maintaining identical configurations but transitioning to the ResNet50 model resulted in a notable decrease in accuracy, plummeting to approximately 48%. This decline suggests subpar performance of the model in signal classification tasks.

Confusion Matrix: The confusion matrix specific to the ResNet50 model is visualized in Figure 13.

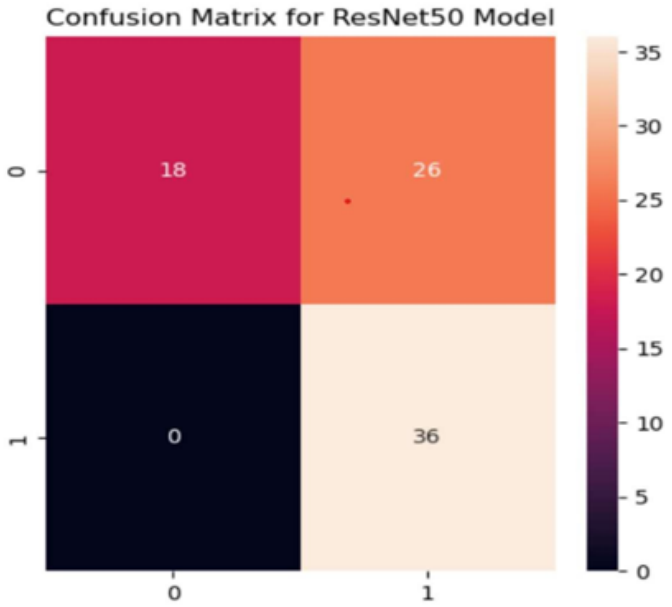

Figure 13: Confusion Matrix for ResNet50

Performance under induced noise levels: The model underwent training and testing under consistent noise levels. Figure 14 illustrates the model's performance in this regard.

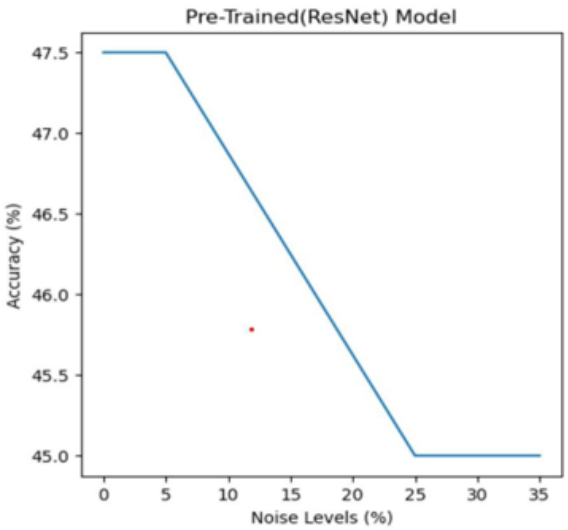

Figure 14: Performance of ResNet50 under different Noise Levels

C. Pre-trained VGG16

Accuracy: Upon training and testing the dataset using the pre-trained VGG16 model, we achieved an accuracy of 86.25%. This model exhibited notably superior performance compared to the previous one.

Confusion Matrix: The confusion matrix corresponding to the VGG16 model is visualized in Figure 15.

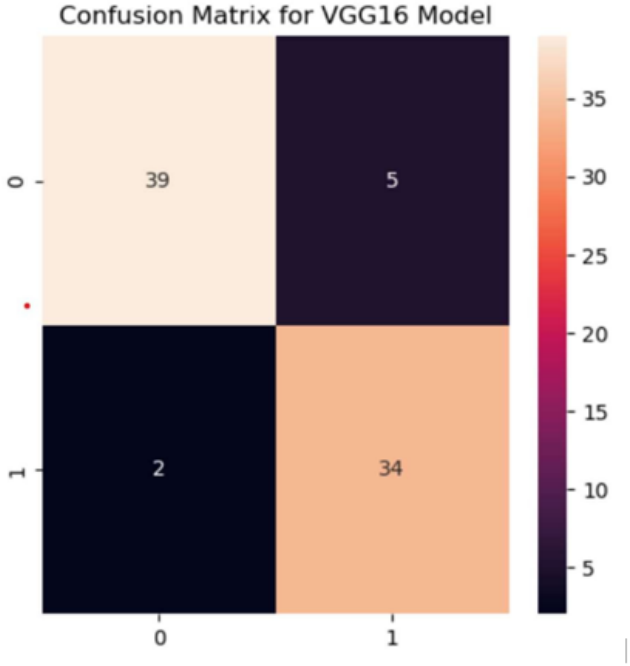

Figure 15: Confusion Matrix for VGG16

Performance under induced noise levels: The model underwent rigorous testing across various noise levels. Below, in Figure 16, are the performance results.

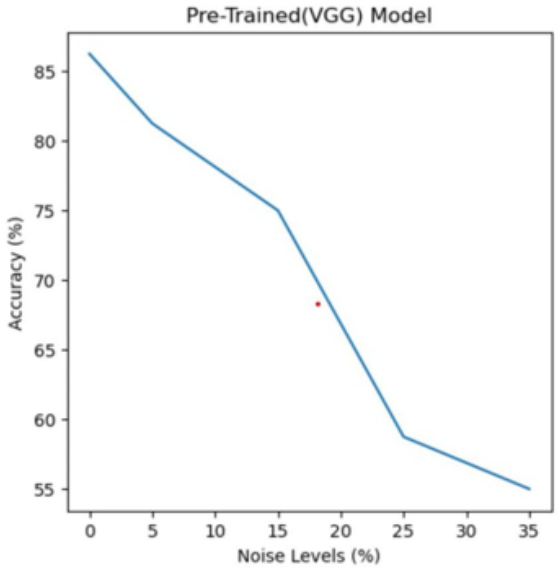

Figure 16: Performance of VGG16 under different Noise Levels

5. CONCLUSION

Through a series of experiments, we thoroughly examined the performance of various models for audio signal recognition.

One experiment involved convolving a signal with its noisy counterpart, revealing a unimodal peak. Additionally, we observed bimodal peaks when convolving a noisy signal with another. These observations yield valuable insights into the behavior of audio signals in noisy environments and effective processing techniques.

Furthermore, we conducted a comparative analysis between a custom CNN model and transfer learning models utilizing ResNet50 or VGG16 as base models. Interestingly, our findings indicate that the custom CNN model surpassed the transfer learning models in terms of accuracy, as illustrated in Figure 17.

The custom CNN model exhibited higher initial accuracy, with a slower decline compared to the transfer learning models, as evidenced in Figure 18. These results underscore the efficacy of the custom CNN approach for audio signal recognition tasks.

Noise Level	Accuracy with Custom Model	Accuracy with VGG16	Accuracy with ResNet50
0	100	86.25	47.5
5	100	81.25	47.5
15	100	75	46.25
25	78.75	58.75	45
35	55	55	45

Figure 17: Reported Accuracies for all the models

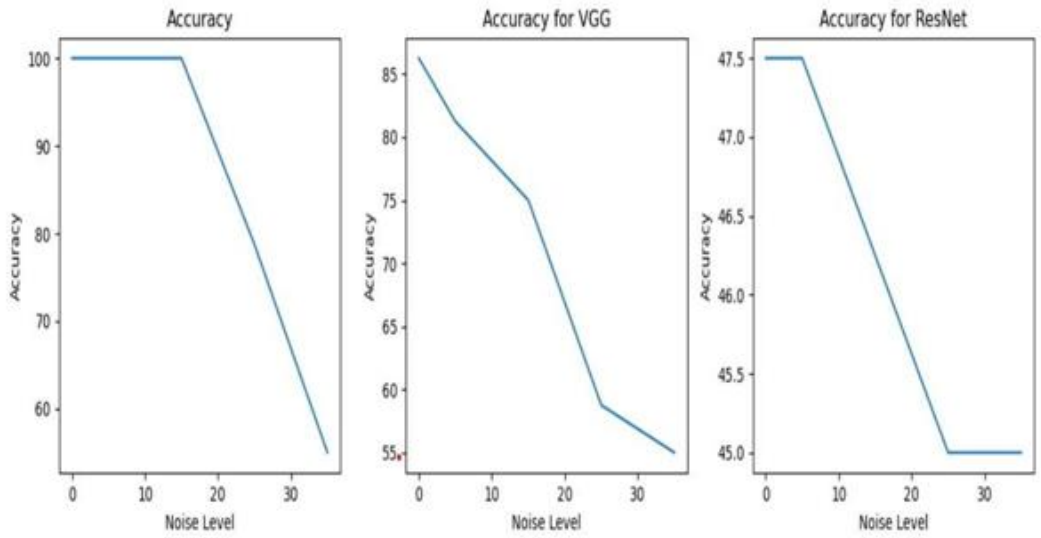

Figure 18: Performance view of the models under noise

Indeed, it's remarkable that the accuracy of the custom CNN model experienced minimal decline when trained on a small dataset. This resilience suggests that the model's generalization capabilities were preserved, even with limited data. Such a promising outcome indicates the potential effectiveness of the model, even in scenarios where data availability is constrained.

In summary, our project has yielded valuable insights into the performance of various models for emergency audio recognition. These findings hold significant implications for the development of more accurate and efficient audio recognition systems, particularly in real-world emergency scenarios. By leveraging these insights, we can advance the development of robust systems capable of accurately identifying emergency audio signals, ultimately contributing to enhanced emergency response efforts and public safety.

6. REFERENCES

[1] https://www.kaggle.com/datasets/ vishnu0399/emergency-vehicle-siren-sounds.

[2] https://www.researchgate.net/publication/345144391_Em ergency_Detection_using_audio

[3] https:/www.researchgate.net/publication/320822533_Det ection_of_alarm_sounds_in_noisy_environments

[4] https://www.ccee.ncsu.edu/ccli-sensors/wp- content/uploads/sites/4/2015/09/Module-4.pdf

[5] https://www.analyticsvidhya.com/blog/2022/03/impleme nting-audio-classification-project-using-deep-learning/

[6] https://medium.com/x8-the-ai-community/audio- classification-using-cnn-coding-example-f9cbd272269e

[7] https://realpython.com/python-scipy-fft/

[8] https://towardsdatascience.com/cnns-for-audio- classification-6244954665ab

[9] https://towardsdatascience.com/convolutional-neural- networks-explained-9cc5188c4939

[10] https://paperswithcode.com/method/max-pooling

[11]https://keras.io/api/layers/regularization_layers/dropout/