PES UNIVERSITY

(Established under Karnataka Act No. 16 of 2013)

100-ft Ring Road, Bengaluru – 560 085, Karnataka, India

Audio Technology Report

on

Detection and Classification of Emergency Sirens in Traffic

Submitted by

Sumukh S (PES1UG21EC302)
Sushanth M G (PES1UG21EC305)
V Shankar (PES1UG21EC318)

June - July 2023

under the guidance of

Dr. Ashwini
Associate Professor
Department of ECE
PES University

FACULTY OF ENGINEERING

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

PROGRAM B.TECH

CERTIFICATE

This is to certify that the Report entitled

Detection and Classification of Emergency Sirens in Traffic

is a bona fide work carried out by

Sumukh S (PES1UG21EC302)

Sushanth M G (PES1UG21EC305)

V Shankar (PES1UG21EC318)

In partial fulfilment for the completion of course work in the Program of Study B.Tech in Electronics and Communication Engineering, under rules and regulations of PES University, Bengaluru during the period June - July 2023. It is certified that all corrections/suggestions indicated for internal assessment have been incorporated in the report. The report has been approved as it satisfies the academic requirements in respect of Audio Technology Project Work.

Signature with date & Seal
(Dr. Ashwini)
Guide

Signature with date & Seal
Dr. Anuradha M
Chairperson

Name and signature of the examiners:

1.

2.

DECLARATION

We, Sumukh S, Sushanth MG and V Shankar hereby declare that the report entitled "**Detection and Classification of Emergency Sirens in Traffic**" is an original work done under the guidance of Dr. Ashwini, Associate Professor in the Dept of ECE and is being submitted as a partial requirement for completion of Audio Technology Project Work of the B.Tech ECE Program of study during June-July 2023.

Place : PES University

Date : 13 -07-2023

Name of the student	Signatures
1.Sumukh S (PES1UG21EC302)	
2. Sushanth MG (PES1UG21EC305)	
3.V Shankar (PES1UG21EC318)	

INDEX

INTRODUCTION

Machine Learning Models Used To Train And Test The Dataset

Convolutional neural networks

Convolutional neural networks are distinguished from other neural networks by their superior performance with image, speech, or audio signal inputs. They have three main types of layers, which are the Convolutional layer, Pooling layer and the Fully-connected (FC) layer.

The convolutional layer is the first layer of a convolutional network. While convolutional layers can be followed by additional convolutional layers or pooling layers, the fully-connected layer is the final layer.

With each layer, the CNN increases in its complexity, identifying greater portions of the image. Earlier layers focus on simple features, such as colours and edges. As the image data progresses through the layers of the CNN, it starts to recognize larger elements or shapes of the object until it finally identifies the intended object.

Convolutional neural networks power image recognition and computer vision tasks.

LSTM (Long short-term memory networks)

Recurrent Neural Networks (RNNs) and Long Short-Term Memory (LSTM) networks are powerful architectures for processing sequential data.

RNNs are a class of neural networks specifically designed to handle sequential data by introducing recurrent connections that allow information to flow from one step to the next. They maintain an internal hidden state that captures the network's memory of previous inputs. RNNs are capable of capturing temporal dependencies in data, making them suitable for tasks such as natural language processing, speech recognition, and time series analysis. However, traditional RNNs suffer from the vanishing gradient problem, which limits their ability to learn long-term dependencies.

To address this issue, LSTM networks were introduced. LSTMs are a type of RNN architecture that incorporates memory cells and gating mechanisms to selectively store and retrieve information over long sequences. LSTMs mitigate the vanishing gradient problem by allowing the network to retain information for extended periods. This is achieved through three main gates: the input gate, the forget gate, and the output gate. These gates regulate the flow of information and selectively update the memory cells based on the current input and the previous memory state.

The input gate controls the amount of new information to be stored in the memory cells, the forget gate determines which information to discard from the cells, and the output gate regulates the amount of information to output to the next step. By effectively managing information flow, LSTMs are capable of capturing and modelling long-term dependencies in sequential data.

XGBoost

XGBoost (Extreme Gradient Boosting) is a powerful and popular machine learning algorithm that belongs to the boosting family. It is known for its ability to deliver highly accurate predictions and handle diverse types of data, making it widely used in various domains and competitions.

XGBoost builds an ensemble of weak prediction models, typically decision trees, in a sequential manner. Each subsequent model is trained to correct the errors made by the previous models. This boosting process creates a strong predictive model by combining the individual predictions of multiple weak models.

The key strength of XGBoost lies in its optimization objectives and regularization techniques. It employs a gradient boosting framework, which minimizes a user-defined loss function by iteratively adding new models to the ensemble. XGBoost places a high emphasis on the reduction of both bias and variance to achieve better generalization performance.

To further enhance the model's performance, XGBoost incorporates feature importance pruning. Feature importance pruning helps identify and discard less relevant features, allowing the algorithm to focus on the most informative ones.

AdaBoost

AdaBoost (Adaptive Boosting) is an ensemble learning algorithm that aims to improve the performance of weak learners by iteratively combining their predictions. It is widely used in classification tasks and has been proven effective in various domains.

The main idea behind AdaBoost is to assign weights to each instance in the training set, with higher weights given to misclassified instances. The algorithm then trains a weak learner on this weighted data and computes its performance. Subsequently, it adjusts the weights of the misclassified instances, placing more emphasis on them in the next iteration. This iterative process continues, and the final model is formed by combining the predictions of all weak learners, weighted by their accuracy.

AdaBoost excels in handling complex datasets and capturing intricate decision boundaries. By focusing on the challenging examples, it is able to create a strong ensemble model that can generalize well to unseen data. Additionally, AdaBoost's simplicity and interpretability make it popular among practitioners.

However, AdaBoost is sensitive to noisy data and outliers, as the algorithm assigns higher weights to misclassified instances, potentially amplifying the impact of outliers. Careful preprocessing and handling of outliers are essential to mitigate this issue.

XGBoost (Extreme Gradient Boosting) and AdaBoost (Adaptive Boosting) are both ensemble learning algorithms that combine weak learners to create strong models. However, they differ in several key aspects.

- *Optimization: XGBoost uses gradient boosting to minimize a user-defined loss function, iteratively adding weak learners. AdaBoost adjusts instance weights to emphasize misclassified samples in each iteration.*

- *Weak Learners: XGBoost typically employs decision trees as weak learners, while AdaBoost can work with any weak learner, such as decision stumps or other algorithms.*
- *Handling Misclassified Instances: XGBoost minimizes overall loss using gradients without specifically assigning higher weights to misclassified instances. AdaBoost increases the weights of misclassified instances to focus on them in subsequent iterations.*
- *Regularization Techniques: XGBoost incorporates regularization techniques like shrinkage and feature importance pruning. AdaBoost indirectly performs regularization through weight adjustments but lacks explicit regularization mechanisms.*
- *Performance: XGBoost excels in scalability, speed, and handling large datasets efficiently. AdaBoost handles complex datasets and improves performance by focusing on challenging examples.*

Support Vector Machines

SVMs aim to find the best hyperplane in a high-dimensional feature space that can separate different classes of data. The main idea behind SVM is to transform the input data into a higher-dimensional space using a kernel function. In this transformed space, SVM finds the hyperplane that maximizes the margin, which is the distance between the hyperplane and the nearest data points of each class. This margin maximization ensures a robust separation between classes and promotes generalization to unseen data.

SVMs are particularly effective in handling high-dimensional data and can handle both linearly separable and non-linearly separable cases.

One advantage of SVMs is their ability to handle data points that lie close to the decision boundary, known as support vectors. These support vectors play a crucial role in defining the hyperplane and making predictions. SVMs can generalize well, even with a relatively small number of support vectors.

Furthermore, SVMs can utilize different types of kernel functions, such as linear, polynomial, Gaussian (RBF), or sigmoid functions. The choice of the kernel depends on the data and the complexity of the problem, allowing SVMs to capture complex relationships between features.

However, SVMs may be sensitive to the choice of hyperparameters, such as the kernel type and regularization parameter. Additionally, SVMs can be computationally expensive for large datasets

Random Forests

The core idea behind Random Forest is to construct multiple decision trees and combine their predictions to make a final prediction. Each decision tree is trained on a random subset of the training data and a random subset of features. This randomization helps to reduce overfitting and improve the generalization ability of the model.

Random Forest leverages the concept of "bagging" (bootstrap aggregating) to create diverse decision trees. Bagging involves creating multiple subsets of the training data through random sampling with replacement. By

training each decision tree on a different subset of the data, Random Forest promotes diversity among the trees and reduces the variance of the model.

During the prediction phase, each decision tree in the Random Forest independently makes a prediction, and the final prediction is determined by majority voting (for classification) or averaging (for regression) over all the individual tree predictions.

Random Forest offers several advantages, including robustness against overfitting, good performance on large and high-dimensional datasets, and the ability to handle missing values and outliers. It is also relatively insensitive to hyperparameter tuning and provides built-in measures of feature importance. However, Random Forest models can be computationally expensive to train and require more memory compared to individual decision trees.

Features Extracted From .wav Audio Files

Root Mean Squared Energy:

Root-mean-square (RMS) energy is a measure of the energy in a signal, often used in audio signal processing. It is defined as the square root of the mean of the squared values in a signal. The python library librosa provides a function librosa.feature.rms() for computing the RMS value for each audio frame in a signal. One common application of RMS energy is in determining the loudness of an audio signal. Since the RMS energy provides a measure of the overall energy in a signal, it can be used as an indicator of the perceived loudness of the signal.

Mel Frequency Cepstral Coefficients:

They give information on the envelope of the spectrum of the audio signal. Mel-frequency Cepstral Coefficients are obtained by first converting the audio signal into a sequence of frames, which are then transformed into the frequency domain using the Fourier Transform. The resulting power spectrum is then passed through a Mel filterbank, which integrates the power in different frequency bands to better match human perception of sound. Finally, the logarithm of the resulting Mel-filterbank energies is computed and transformed using the Discrete Cosine Transform (DCT), resulting in a set of typically 12-39 MFCC coefficients that can be used as features for speech recognition or other processing applications.

Spectral Centroid:

Spectral Centroid represents the center of mass of the spectrum and provides information about the "brightness" of the audio. It is strongly associated with the perceived brightness of the sound . The Spectral Centroid can be used to distinguish between different types of sounds, such as musical instruments or speech sounds.

Spectral Bandwidth:

Spectral Bandwidth measures the range of frequencies in the audio signal.

Zero Crossing Rate:

Zero Crossing Rate is a measure used in signal processing to quantify the number of times a signal waveform crosses the zero axis ie the number of times a signal waveform goes from the positive axis to the negative axis or vice versa. ZCR has been used in separating the voiced and unvoiced parts of speech from a signal.

Chroma Features:

Chroma Feature represents the pitch content of audio by dividing the octave into 12 equal-sized bins. A chroma feature relies on mapping all pitches from a signal to the twelve pitch-classes of the chromatic scale. The chromatic scale consists of the twelve musical notes that are equally spaced within an octave.

Different methods exist to calculate chroma features, but they typically involve computing a harmonic pitch-class profile(frequency distribution over each pitch class) based on the magnitude of the short-time Fourier transform (STFT) or constant-Q transform (CQT). Next, the pitch-class profile is mapped to a chroma feature vector, where each element of the vector represents the energy or frequency content of one of the 12 pitch-classes.

Parameters Used In The Comparison Of Machine Learning Models

Precision regards the quality of a positive prediction made by a Machine Learning model. Precision is calculated by taking the ratio of true positive predictions to the total number of positive predictions made by the model, which means it measures the ratio of accurately predicted positive instances. However, a high precision score, in some cases, may come with a lower recall score. This is due to a trade-off when making predictions, between keeping false positives to a minimum and capturing all possible positive samples .

Recall refers to the proportion of actual positive instances that are correctly identified by a machine learning model. It is also known as the true positive rate (TPR), and it is calculated as the ratio of true positive predictions to the total number of actual positive instances. Recall is commonly used as a performance metric when evaluating classifiers, particularly in situations where detecting all positives is more important than avoiding false positives.

F1 score is a common evaluation metric used in machine learning to measure the accuracy and reliability of a model's predictions. It provides a more balanced view of a model's performance than using precision or recall alone. The F1 score is a harmonic mean of precision and recall, with values ranging from 0 to 1. A perfect score of 1 indicates that a model has achieved perfect precision and recall, which means that all of its predictions are

correct and it has found all relevant examples. The F1 score can be calculated using the formula: *2 * ((precision * recall) / (precision + recall)).*

Support refers to the number of occurrences of each class in the specified dataset. It is used to calculate various performance metrics such as precision, recall, and F1 score. Specifically, support is the sum of true positives and false negatives for each class. Support is useful especially when dealing with imbalanced datasets, where one class has significantly more samples than the other classes.

Macro Average is a method used to calculate the overall performance of a model across all predicted classes. Macro average computes the metric independently for each class and then takes the average of the per-class metric scores. This means that each class is weighted equally, regardless of its frequency or number of samples in the dataset. Hence, macro average ensures that the evaluation metric is not influenced by class imbalance.

Weighted average is a type of average calculation that takes into account the relative importance or frequency of some factors in a data set. In a weighted average, each data point value is multiplied by the assigned weight, which is then summed and divided by the sum of the weights. The use of weighted averages can improve the accuracy of the data analysis, as it takes into account the varying degrees of importance of the numbers in a dataset.

LITERATURE SURVEY

[1] Asif M, Usaid M, Rashid M, Large-scale audio dataset for emergency vehicle sirens and road noises, Sci Data 9, 599 (2022).

In this research work, the development of the acoustic-based vehicle type classification system is utilized the Mel-Frequency Cepstral Coefficient (MFCC), its Deltas and Delta-Deltas as the main feature extraction method and a Multi-Layer Feedforward Neural Network (MLFFNN) for the classification on passing vehicle sounds. The system is able to objectively recognize the type of vehicles such as bus, car, motorcycle, and truck passing on the highway road with the specific acceleration.

From the experimental results, it is inferred that the type and distance of moving vehicle can be identified using the Multi-Layer Feedforward Neural Network classifier. The performance of the system is obtained by 82.0% in the analysis of the experimentation.

[2] Mittal U, Chawla P, Acoustic Based Emergency Vehicle Detection Using Ensemble of Deep Learning Models. Procedia Comput. Sci. 2023, 218, 227–234.

This research paper make use of MFCC Feature, compares 3 different deep Learning models, which are Dense Layer, CNN and RNN and comes to the conclusion that deep learning is most effective for the Google Audio Dataset and it has an accuracy of about 98.70%.

Performance analysis of deep learning models is done with various machine learning models like Perceptron, SVM, decision tree etc. Proposed model can be implemented in real time for detecting emergency vehicles approaching towards the intersection and priority can be provided to reduce their waiting time.

[3] Suhaimy, M. A., Halim, I. S. A., Hassan, S. L. M., and Saparon, A. (2020, December). Classification of ambulance siren sounds with MFCC SVM. AIP Conference Proceedings (Vol. 2306, No. 1, p. 020032). AIP Publishing LLC.

Important audio features have been extracted from raw data and passed into extreme learning machines (ELM) for training. ELMs have been used in this work because of its simplicity and shorter run-time. The model has been trained on acoustic data by extracting features from audio recording and then fed into a model for classification, and MFCC as well as Zero crossings have been extracted. Five different models were trained alongside ELM such as K-Nearest Neighbours (KNN), Random Forest (RF), Support Vector Machine (SVM), Multilayer Perceptron (MLP) and Convolutional Neural Network (CNN)

With ELMs the accuracy can reach up to 97% that is comparable to the current state of the art while dramatically reducing the training time. Moreover ELMs are 10 times faster than KNN and over 4000 times faster than state of the art CNN or MLP models. The use of zero crossing in this model is a unique feature.

METHODOLOGY

Problem Statement: Develop an efficient and accurate system for audio detection and classification of emergency sirens from traffic noise to enhance road safety and emergency response to civil distress.

In this study, we assembled a final dataset consisting of audio samples from various online resources, collected through internet sources, such as Kaggle and AudioSet offered by Google. The dataset comprises of six classes: traffic, ambulance, firetruck, police, weather alert (or civil defense) siren and fire alarm, with each class containing a total of 200 audio data files, except for the police class, which has 227 samples. All audio files in the dataset are in the .wav format and have a duration of 3 seconds.

To develop effective machine learning models, we explored four different approaches: Convolutional Neural Network (CNN), Long Short-Term Memory (LSTM), Random Forest, and Support Vector Machine (SVM).

For the CNN model, we converted the .wav audio files into .png image files representing their respective spectrograms. This conversion was performed as CNNs excel in image processing tasks. The spectrograms were then utilized as input for the CNN model.

For LSTM, we extracted Mel-Frequency Cepstral Coefficients (MFCC) from the audio samples and stored them in a .csv file. These MFCC values were subsequently employed as input features for the model.

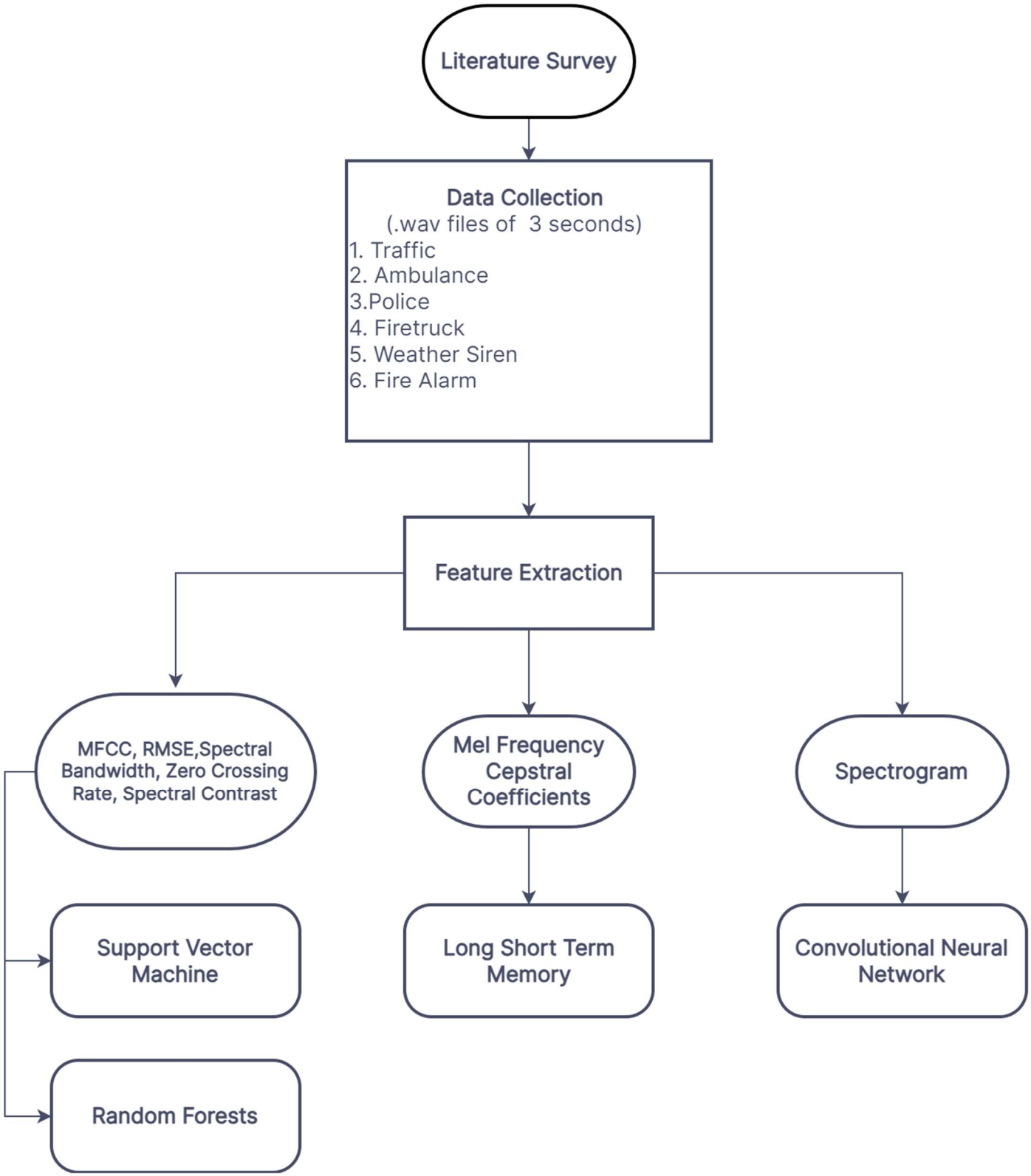
For SVM and Random Forest, Root Mean Square Energy, Zero Crossing Rate, Spectral Centroid, Spectral Bandwidth and Spectral Contrast features were also extracted from the audio files, along with MFCC.

Furthermore, we employed XGBoost and AdaBoost algorithms to enhance the performance of the models. Specifically, XGBoost was employed to eliminate the least important features, thereby optimizing the model's performance.

Finally, we evaluated the performance of each model based on various metrics, such as accuracy, precision, recall, support, macro average, weighted average and F1 score. The results were compared among the models to determine their effectiveness in classifying the audio samples into the six defined classes.

By employing different machine learning techniques and comparing their performances, this study provides valuable insights into the effectiveness of CNN, LSTM, Random Forest, and SVM models for audio classification tasks. The findings contribute to the development of reliable and accurate systems for real-world applications such as traffic monitoring and emergency response to civil distress.

BLOCK DIAGRAM



SOFTWARE USED

- ***Google Collab*** is a cloud-based development environment that allows users to write and execute Python code, collaborate with others, and leverage powerful libraries and frameworks for machine learning and data analysis.

- ***NumPy*** is a fundamental package for scientific computing in Python, providing powerful tools for creating and manipulating multidimensional arrays, performing mathematical operations, and integrating with other libraries and tools.

- ***Pandas*** is a popular data manipulation and analysis library in Python, offering a wide range of data structures and functions to efficiently handle and process structured data, perform data cleaning, transformation, aggregation, and exploration tasks.

- ***Matplotlib*** is a comprehensive plotting library in Python, enabling the creation of a wide variety of static, animated, and interactive visualizations for data analysis, presentation, and communication purposes.

- ***Scikit-learn*** is a versatile machine learning library in Python, offering a rich set of algorithms and tools for tasks such as classification, regression, clustering, dimensionality reduction, model selection, and evaluation, along with utilities for data preprocessing and feature engineering.

- ***Librosa*** is a Python library for audio and music analysis, providing a wide range of functions and tools to extract various audio features, perform signal processing operations, visualize spectrograms, and work with audio data in machine learning applications.

- ***Seaborn*** is a data visualization library built on top of Matplotlib, providing a high-level interface to create attractive and informative statistical graphics in Python, with built-in support for statistical estimation and visualization of distributions, relationships, and patterns in data.

- ***Torch*** is a scientific computing framework in Python, offering support for tensor computation with GPU acceleration, automatic differentiation for building and training neural networks, and a rich ecosystem of libraries and tools for machine learning research and development.

- ***PyTorch Lightning*** is a lightweight PyTorch wrapper that simplifies the training and development of deep learning models, providing a high-level interface for handling common tasks such as data loading, model configuration, training loops, and experiment management, with built-in support for distributed training and advanced optimizations.

- ***PIL (Python Imaging Library)*** is a library for image processing and manipulation in Python, providing functions to open, save, and manipulate various image formats, perform basic and advanced image operations such as resizing, cropping, filtering, and blending, and create custom image effects and transformations.

- ***SciPy*** is a library for scientific and technical computing in Python, providing a wide range of functions and tools for numerical integration, optimization, linear algebra, signal and image processing, statistics, and other scientific computing tasks, building upon NumPy for efficient array operations.

- ***TensorFlow*** is an open-source machine learning framework developed by Google, offering a flexible and scalable ecosystem for building, training, and deploying machine learning models across different platforms and devices, with extensive support for deep learning and distributed computing.

- ***Keras*** is a high-level neural networks API in Python, providing a user-friendly interface to build, train, and deploy deep learning models, with support for various backends such as TensorFlow, Theano, and Microsoft Cognitive Toolkit, making it accessible to both beginners and advanced users.

- ***Pickle*** is a Python module that enables object serialization and deserialization, allowing users to save Python objects to disk and load them back into memory, providing a convenient way to store and exchange complex data structures, models, and configurations.

- ***AudioSet*** is a large-scale dataset created by Google, consisting of labeled audio segments from a wide range of real-world sounds, designed to facilitate research and development in audio analysis, classification, and scene understanding tasks.

- ***FFmpeg*** is a powerful multimedia framework and command-line toolset that allows users to decode, encode, transcode, and manipulate audio and video files, providing extensive capabilities for format conversion, editing, streaming, and multimedia processing.

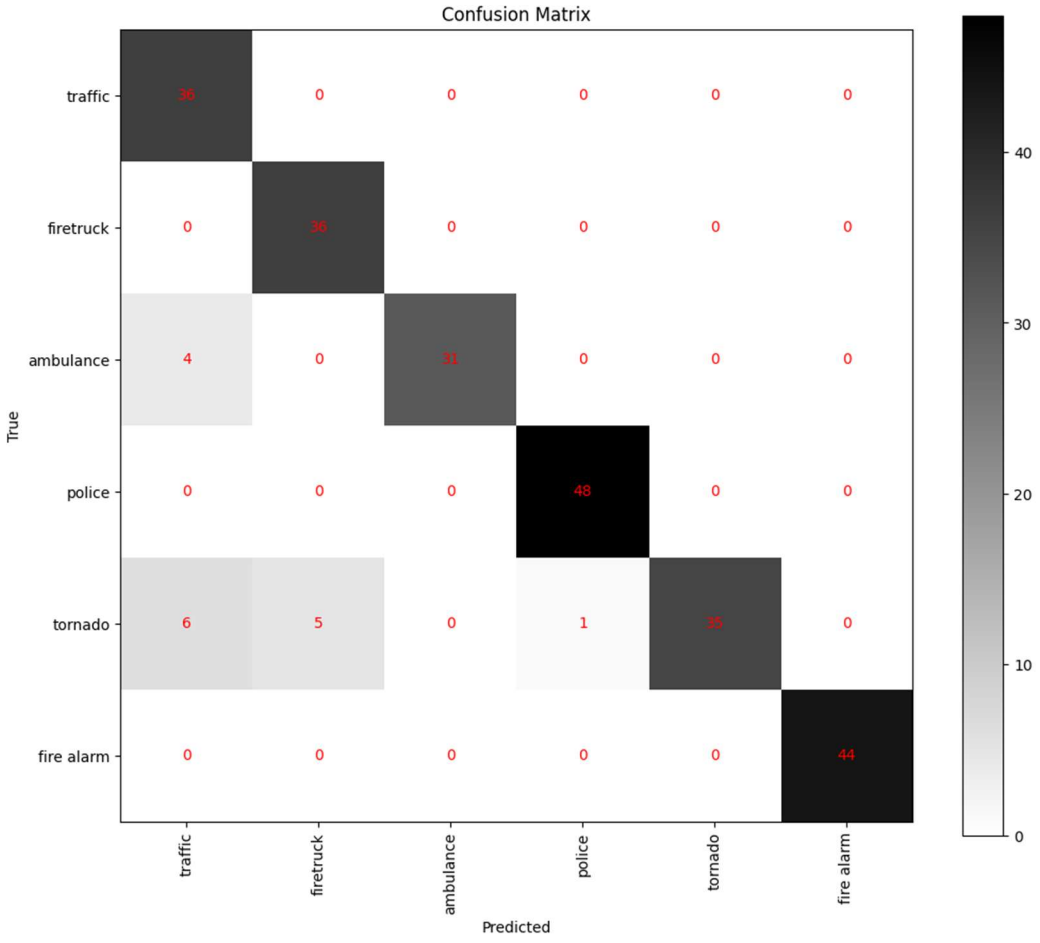
RESULTS AND COMPARISON

Convolutional Neural Network:

Accuracy: 93.49%

	Precision	*Recall*	*F1-Score*	*Support*
Ambulance	0.7826	1.0000	0.8780	36
Firetruck	0.8780	1.0000	0.9351	36
Traffic	1.0000	0.8857	0.9394	35
Police	0.9796	1.0000	0.9897	48
Tornado	1.0000	0.7447	0.8537	47
Fire_Alarm	1.0000	1.0000	1.0000	44
Macro Avg	0.9400	0.9384	0.9326	246
Weighted Avg	0.9464	0.9350	0.9341	246

Confusion Matrix:

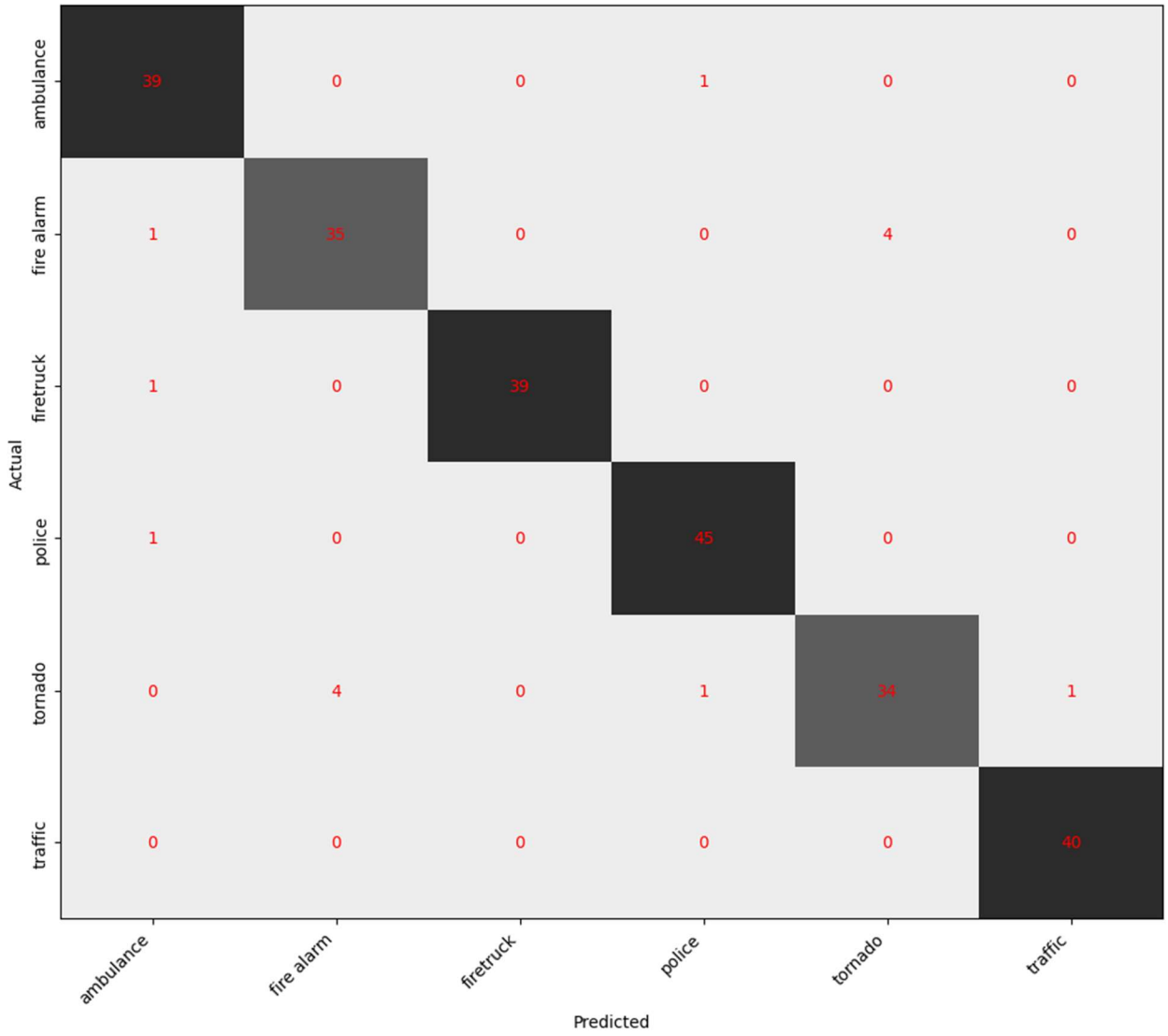


Long Short Term Memory:

Accuracy: 94.31%

	Precision	*Recall*	*F1-Score*	*Support*
Ambulance	0.9300	0.9700	0.9500	40
Firetruck	0.9600	0.8800	0.8900	40
Traffic	1.0000	0.9700	0.9900	40
Police	0.9600	0.9800	0.9700	46
Tornado	0.8900	0.8500	0.8700	40
Fire_Alarm	0.9800	1.0000	0.9900	40
Macro Avg	0.9400	0.9400	0.9400	246
Weighted Avg	0.9400	0.9400	0.9400	246

Confusion Matrix:

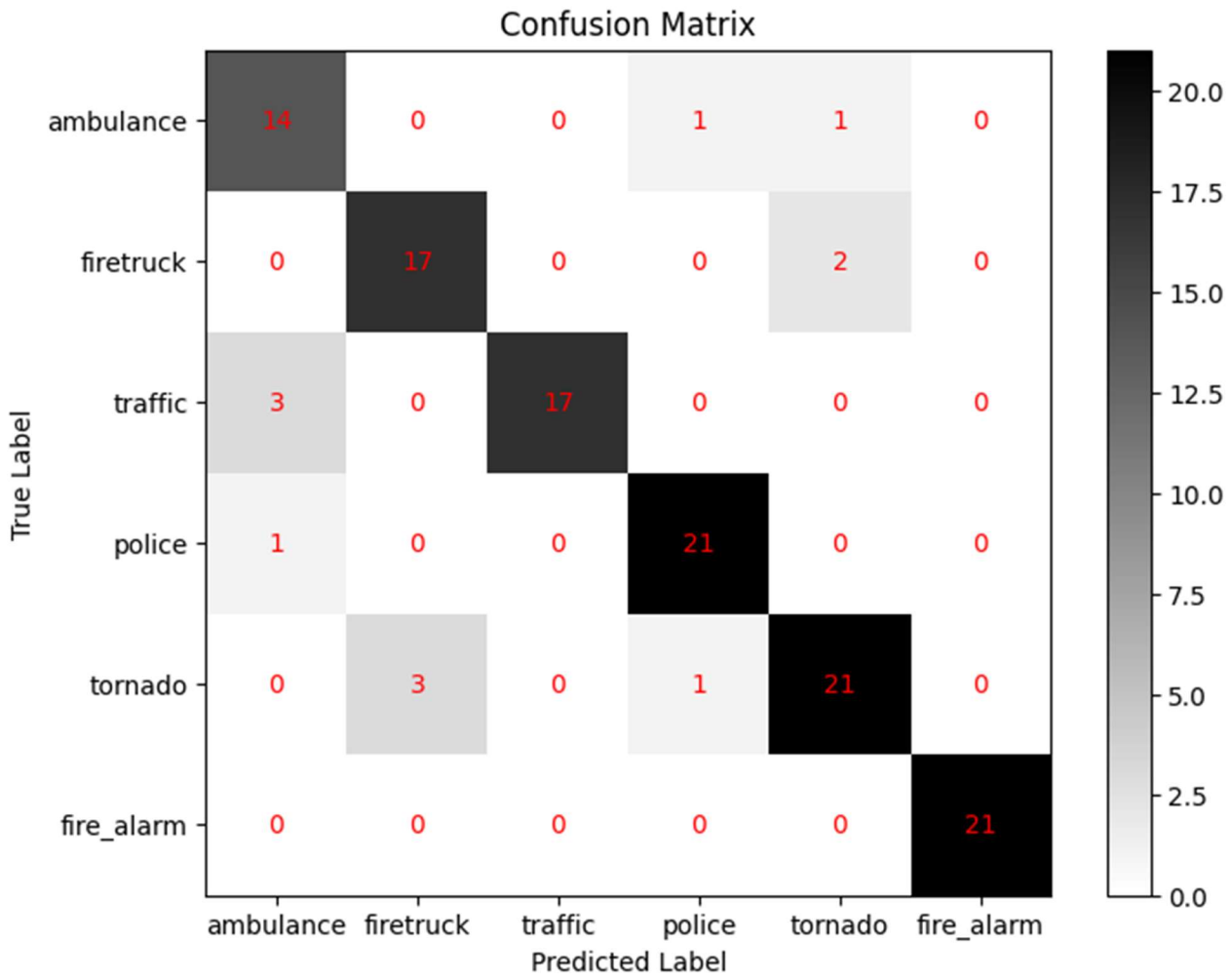


Support Vector Machine:

Accuracy: 90.24%

	Precision	*Recall*	*F1-Score*	*Support*
Ambulance	0.7778	0.8750	0.8235	16
Firetruck	0.8500	0.8947	0.8718	19
Traffic	1.0000	0.8500	0.9189	20
Police	0.9130	0.9545	0.9333	22
Tornado	0.8750	0.8400	0.85710	25
Fire_Alarm	1.0000	1.0000	1.0000	21
Macro Avg	0.9026	0.9024	0.9008	123
Weighted Avg	0.9070	0.9024	0.9231	123

Confusion matrix:

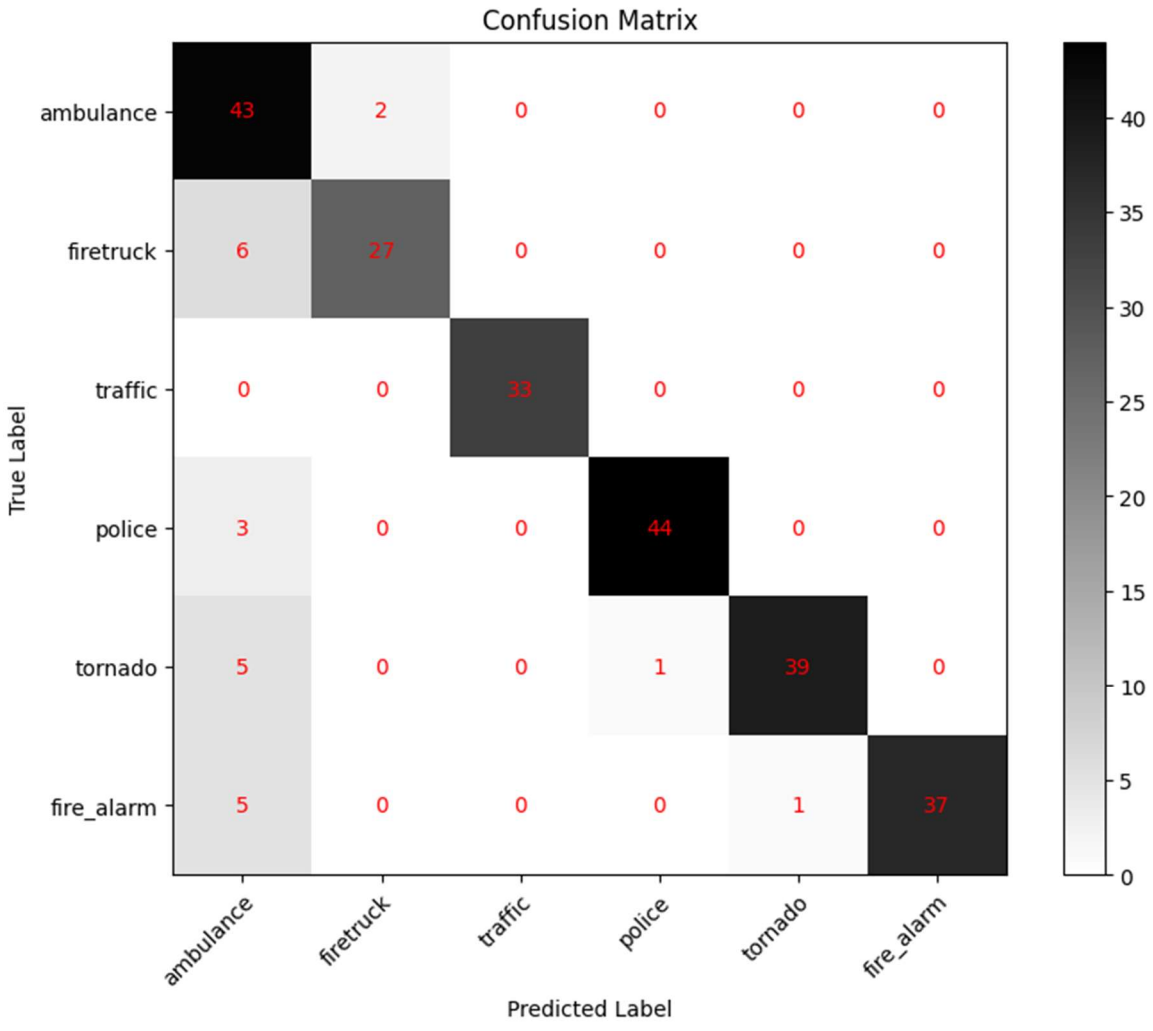


Random Forest:

Accuracy: 87.80 %

	Precision	*Recall*	*F1-Score*	*Support*
Ambulance	1.0000	0.8000	0.8889	45
Firetruck	0.9130	0.8182	0.8710	33
Traffic	1.0000	1.0000	1.0000	33
Police	0.9778	0.9362	0.9565	47
Tornado	0.9750	0.8667	0.9176	45
Fire_Alarm	1.0000	0.8605	0.9250	43
Macro Avg	0.9806	0.8702	0.9265	246
Weighted Avg	0.8780	0.8780	0.8780	246

Confusion Matrix:

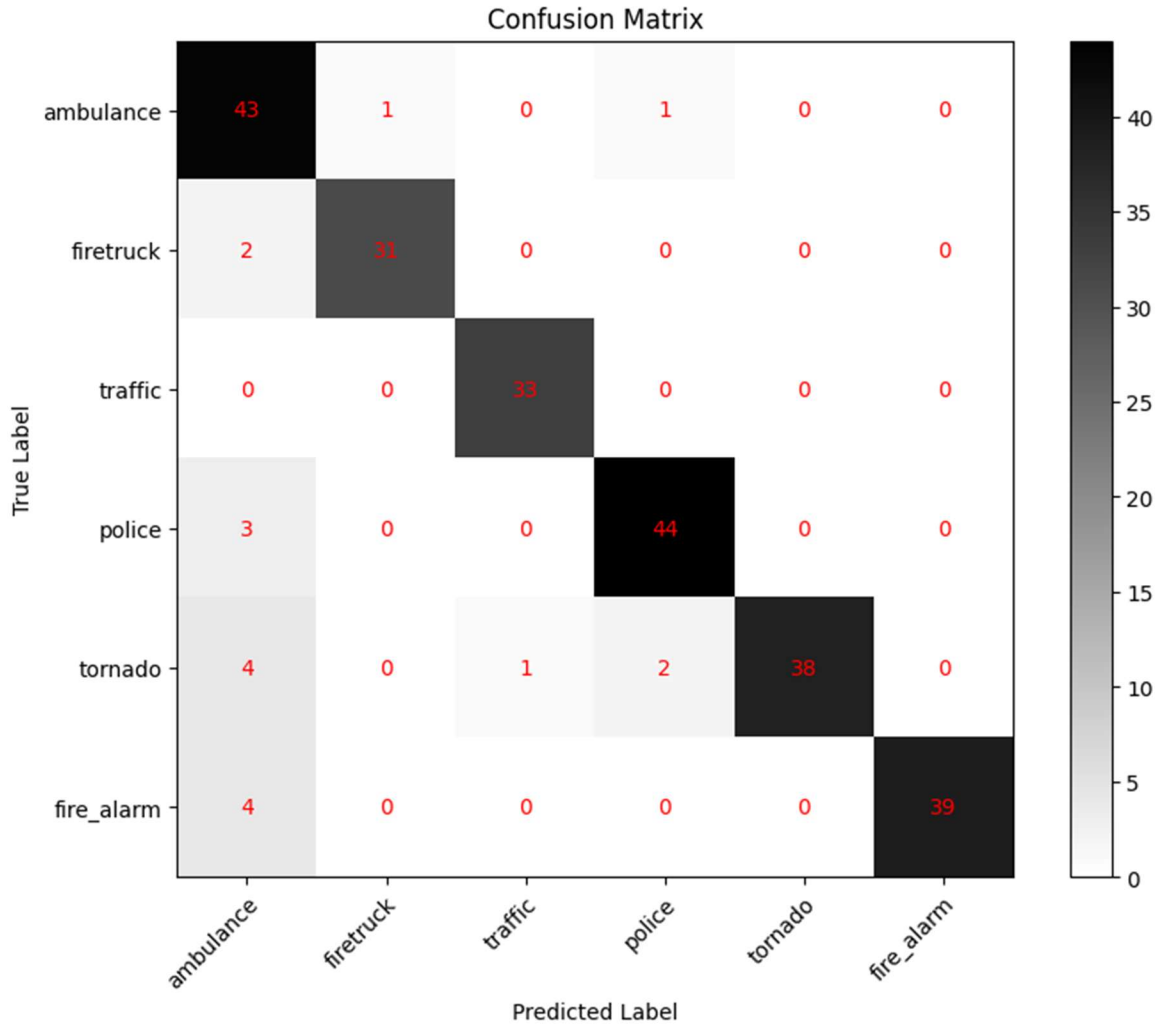


XGBoost:

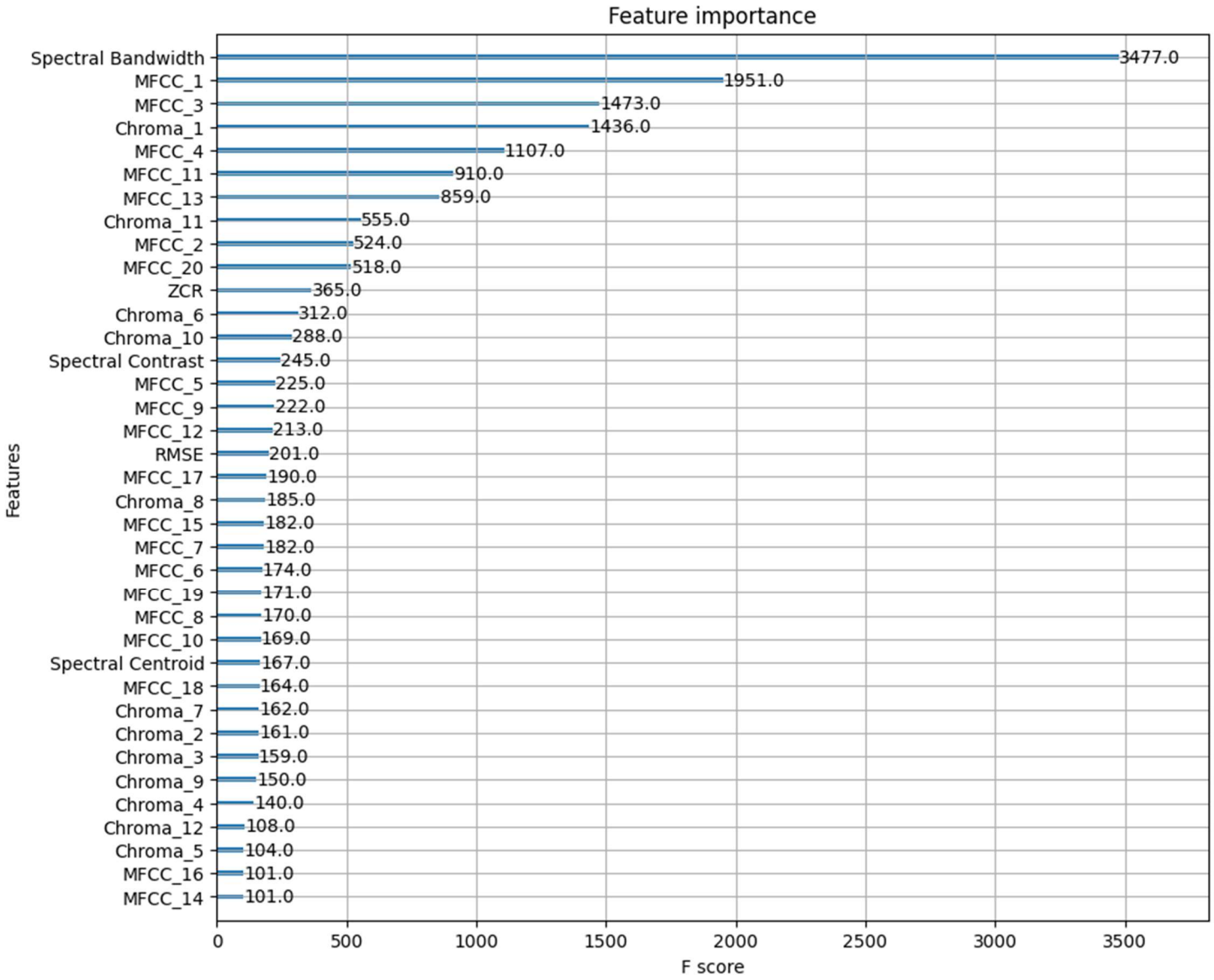
Accuracy: 89.02%

	Precision	*Recall*	*F1-Score*	*Support*
Ambulance	0.9200	0.7800	0.8400	45
Firetruck	0.9400	0.9400	0.9400	33
Traffic	0.9700	1.0000	0.9900	33
Police	0.9400	0.9400	0.9400	47
Tornado	1.0000	08700	0.9300	45
Fire_Alarm	0.9700	0.9100	0.9400	43
Macro Avg	0.9600	0.9000	0.9300	246
Weighted Avg	0.8900	0.9000	0.9000	246

Confusion Matrix:



Graph showing the relative importance of features in classifying the data during training of the XGBoost model:



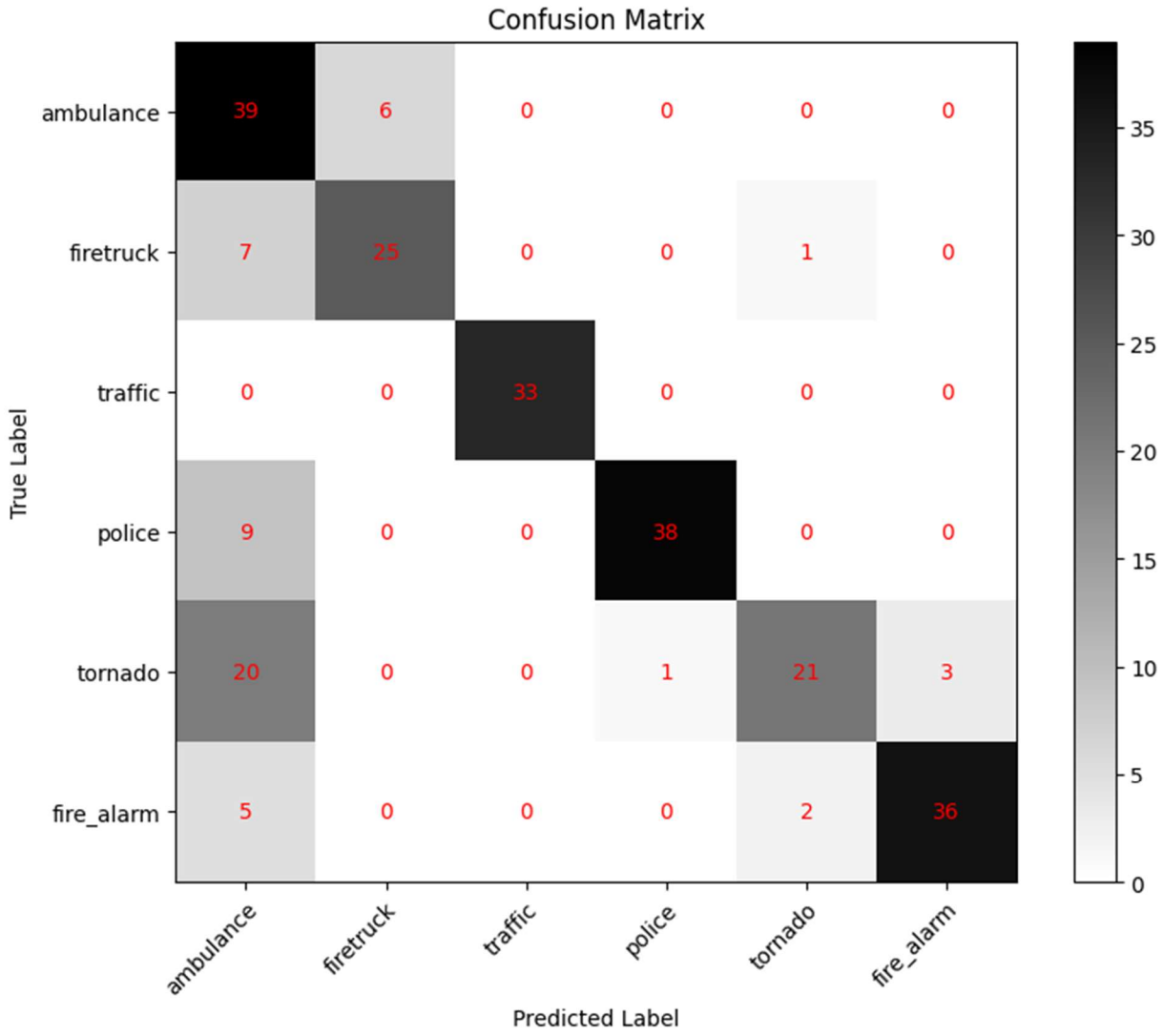
For implementing a Random Forest with a reduced feature set, we consider only the features from 'Spectral Bandwidth' to 'MFCC_20' in the given Bar graph.

Random Forest With Reduced Features:

Accuracy: 74.79%

	Precision	*Recall*	*F1-Score*	*Support*
Ambulance	0.9394	0.6889	0.7949	45
Firetruck	0.8065	0.7576	0.8812	33
Traffic	1.0000	1.0000	1.0000	33
Police	0.9744	0.8085	0.8837	47
Tornado	0.8750	0.4667	0.6087	45
Fire_Alarm	0.9231	0.8372	0.8780	43
Macro Avg	0.9197	0.7598	0.8244	246
Weighted Avg	0.7480	0.7480	0.7480	246

Confusion Matrix:

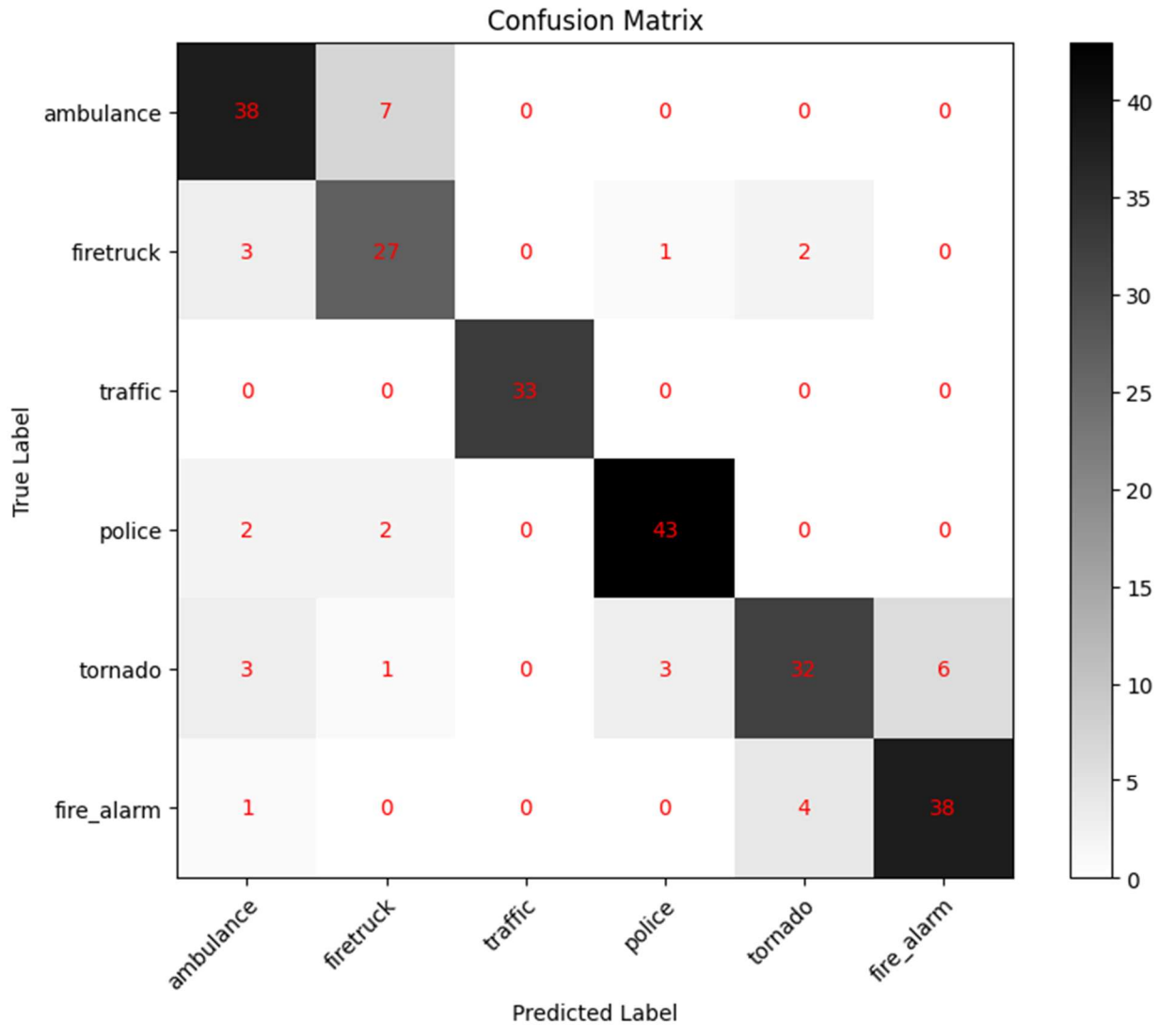


AdaBoost:

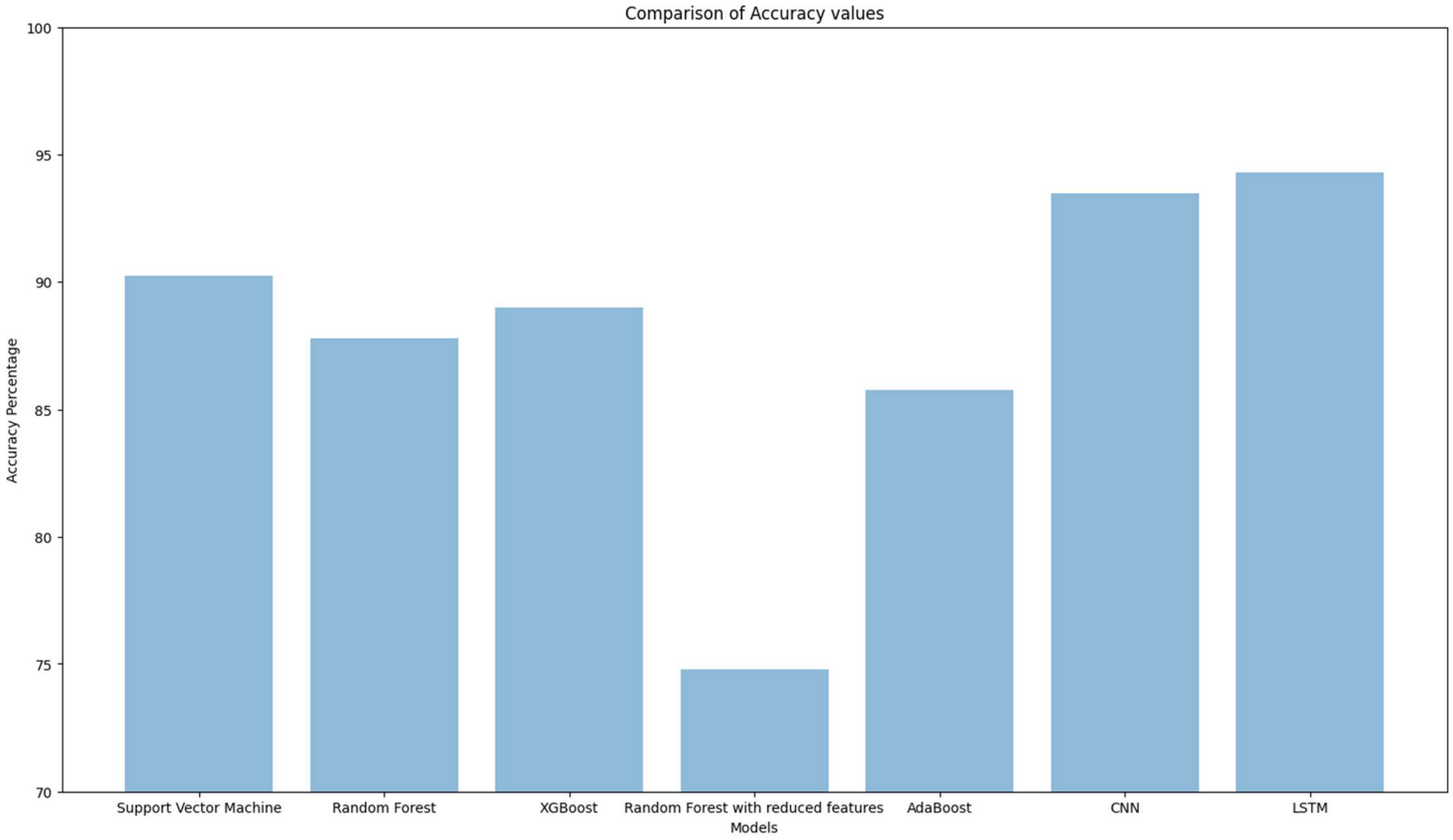
Accuracy: 85.77%

	Precision	*Recall*	*F1-Score*	*Support*
Ambulance	0.8100	0.8400	0.8300	45
Firetruck	0.7300	0.8200	0.7700	33
Traffic	1.0000	1.0000	1.0000	33
Police	0.9100	0.9100	0.9100	47
Tornado	0.8400	0.7100	0.7700	45
Fire_Alarm	0.8600	0.8800	0.8700	43
Macro Avg	0.8600	0.8600	0.8600	246
Weighted Avg	0.8600	0.8600	0.8600	246

Confusion Matrix:



Graph showing comparison of accuracy across various models:



It is observed that LSTM has the highest accuracy. The random forest model with reduced features has the least accuracy. Even though feature optimization leads to lesser computation time, it costs us lots of accuracy.

REFERENCES

[1] Asif M, Usaid M, Rashid M, *Large-scale audio dataset for emergency vehicle sirens and road noises*, Sci Data 9, 599 (2022).

[2] Mittal U, Chawla P, *Acoustic Based Emergency Vehicle Detection Using Ensemble of Deep Learning Models*. Procedia Comput. Sci. 2023, 218, 227–234.

[3] Suhaimy, M. A., Halim, I. S. A., Hassan, S. L. M., and Saparon, A. (2020, December). *Classification of ambulance siren sounds with MFCC SVM*. AIP Conference Proceedings (Vol. 2306, No. 1, p. 020032). AIP Publishing LLC.

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

[5] *https://datagy.io/python-support-vector-machines*

[6] *https://www.datacamp.com/tutorial/cnn-tensorflow-python*

[7] *https://www.alpha-quantum.com/blog/long-short-term-memory-lstm-with-python/long-short-term-memory-lstm-with-python*

[8] *https://stackabuse.com/random-forest-algorithm-with-python-and-scikit-learn*