A project report on

Sound Event Detection in Noisy Environment Using Neural Network

by

AMAN MISHRA (20BCE1662)

ABSTRACT

Sound Event Detection (SED) has emerged as a vital technology within the field of artificial intelligence, enabling the classification of specific acoustic events within complex sonic environments. This paper delves into the challenges and methodologies employed in SED tasks. Traditional approaches to audio analysis relied on dividing sound waves into equal-width frequency bins. However, this linear representation does not align well with human auditory perception. The human ear is more sensitive to variations in lower frequencies compared to higher frequencies. Consequently, a small change in the lower frequency range is perceived as a more significant difference in pitch compared to the same change in a higher frequency range.

To address this limitation, Mel-frequency cepstral coefficients (MFCCs) have been introduced in SED tasks. MFCCs incorporate a Mel scale, which approximates human auditory perception by emphasizing lower frequencies. By transforming raw audio data into MFCCs, a spectral fingerprint of the sound is created. This spectral fingerprint captures the essential characteristics of the sound, allowing for recognition even in noisy environments where the raw audio signal might be significantly distorted.

CONTENTS

LIST OF FIGURES

LIST OF ACRONYMS

ACRONYM	FULL FORM
SED	Sound Event Detection
MFCC	Mel-frequency Cepstral Coefficients
DNN	Deep Neural Network
ReLU	Rectified Linear Unit
FFT	Fast Fourier Transformation

CHAPTER-1

INTRODUCTION

1.1 INTRODUCTION

Sound Event Detection (SED) is a rapidly growing field in machine learning that tackles the challenge of identifying and classifying specific sounds within complex audio recordings, especially those plagued by background noise. The core difficulty lies in separating the target sounds of interest from the ever-present din of environmental noise.

Traditional audio analysis methods relied on dividing sound waves into equal-width frequency bins. However, this approach clashes with how humans perceive pitch. Our ears are more sensitive to changes in lower frequencies compared to higher ones. For instance, a 100 Hz shift in a low rumble would be more noticeable than the same change in a high-pitched squeal.

SED overcomes this limitation by employing techniques like Mel-frequency cepstral coefficients (MFCCs). Inspired by human auditory perception, MFCCs essentially create a spectral fingerprint of the sound, capturing its essence even when obscured by noise. These fingerprints serve as the building blocks for powerful tools like neural networks - complex algorithms that learn from vast amounts of data.

In the context of SED, neural networks are trained on massive datasets of labeled audio clips. Each clip is associated with a specific sound, like a crying baby, a barking dog, or the shattering of glass. By analyzing an immense collection of these MFCC fingerprints alongside their corresponding labels, the neural network develops the ability to associate specific patterns within these fingerprints with their corresponding sounds. This allows the network to effectively classify new, unseen audio data by examining its extracted MFCCs and identifying the most likely sound based on the learned patterns.

SED is a rapidly developing field with immense potential. As the technology matures, it promises to unlock a deeper understanding of the soundscape that surrounds us. We will not just hear the symphony of sounds around us; we will truly understand them,

paving the way for a future that is not only more secure but also richer in information gleaned from the soundscapes of our world.

1.2 OVERVIEW

Sound Event Detection (SED) is a burgeoning area of machine learning focused on identifying and classifying specific sounds within recordings, particularly those brimming with background noise. The core challenge lies in differentiating the target sounds from the ever-present din of the surrounding environment.

Traditional methods for analyzing audio relied on dividing sound waves into segments of equal frequency width. This approach, however, clashes with human perception of pitch. Our ears are more sensitive to variations in lower frequencies compared to higher ones. For example, a slight change in a low-pitched rumble would be more noticeable than the same change in a high-pitched squeal.

SED overcomes this limitation by employing techniques like Mel-frequency cepstral coefficients (MFCCs). Inspired by human hearing, MFCCs essentially create a spectral fingerprint of the sound, capturing its essence even when obscured by noise. These fingerprints become the raw material for powerful tools like neural networks - complex algorithms that learn from vast amounts of data.

In the context of SED, neural networks are trained on massive datasets of labeled audio recordings. Each recording is associated with a specific sound, such as a crying baby, a barking dog, or the shattering of glass. By analyzing a vast collection of these MFCC fingerprints alongside their corresponding labels, the neural network develops the ability to associate specific patterns within these fingerprints with their corresponding sounds. This allows the network to effectively classify new, unseen audio data by examining its extracted MFCCs and identifying the most likely sound based on the learned patterns.

SED is a rapidly developing field with immense potential. As the technology matures, it promises to unlock a deeper understanding of the soundscape that surrounds us. We won't just hear the symphony of sounds around us; we'll truly understand them, paving the way for a future enriched by the information gleaned from the soundscapes of our

world.

1.3 PROBLEM STATEMENT

Traditional audio analysis methods using linear frequency bins struggle with background noise and human auditory perception. This hinders accurate Sound Event Detection (SED) in noisy environments. To address this, SED utilizes Mel-frequency cepstral coefficients (MFCCs). Inspired by human hearing, MFCCs create spectral fingerprints of sounds, capturing their essence even amidst noise.

Machine learning, particularly deep neural networks, play a crucial role. Trained on massive datasets of labeled audio recordings with corresponding sound events (e.g., crying baby, barking dog), these networks learn to associate specific patterns within MFCCs with their corresponding sounds. This allows them to analyze unseen audio data's MFCCs and identify the most likely sound event, achieving higher accuracy in SED tasks.

1.4 SIGNIFICANCE OF STUDY

The significance of this study lies in its potential to overcome the limitations of traditional methods. Dividing sound waves into equal frequency bins, a common practice in earlier audio analysis, fails to account for how humans perceive sound. Our ears are more sensitive to variations in lower frequencies compared to higher ones. This inherent bias can lead to misinterpretations and missed information in SED tasks.

MFCCs offer a promising solution by mimicking human auditory perception. Inspired by the way we hear, MFCCs essentially create a spectral fingerprint of a sound, capturing its essence even when obscured by noise. This study highlights the value of using MFCCs as data for powerful machine learning tools like neural networks.

Neural networks are complex algorithms with the remarkable ability to learn from vast amounts of data. In the context of SED, this study proposes training neural networks on massive datasets of labeled audio recordings. Each recording is associated with a specific sound event, such as a crying baby, a barking dog, or the shattering of glass.

By analyzing a vast collection of these MFCC fingerprints alongside their corresponding labels, the neural network develops the ability to associate specific patterns within these fingerprints with their corresponding sounds.

This approach offers significant advantages for SED. By leveraging MFCCs and the powerful learning capabilities of neural networks, the study paves the way for achieving higher accuracy in SED tasks. This translates to a deeper understanding of the soundscapes around us. Applications range from enhancing security systems by accurately detecting suspicious sounds to environmental monitoring through gunshot or animal call identification. As SED technology matures with the advancements explored in this study, it promises to revolutionize how we interact with and interpret the rich tapestry of sounds that fill our world.

1.5 OBJECTIVES

The primary objective of this research is to investigate and evaluate the effectiveness of Mel-frequency cepstral coefficients (MFCCs) and neural networks in improving the accuracy of Sound Event Detection (SED) tasks, particularly in noisy environments. Traditional audio analysis methods often struggle with background noise and limitations in mimicking human auditory perception. This research aims to address these challenges by exploring two key areas:

Leveraging Mel-Frequency Cepstral Coefficients (MFCCs):
Existing analysis methods typically divide sound waves into equal frequency bins. This approach fails to account for how humans perceive sound. The human ear is more sensitive to variations in lower frequencies compared to higher ones. This inherent bias can lead to misinterpretations and missed information in SED tasks, especially when dealing with noisy environments.
MFCCs offer a promising solution by mimicking human auditory perception. Inspired by the way we hear, MFCCs essentially create a spectral fingerprint of a sound, capturing its essence even when obscured by noise. This objective delves into the application of MFCCs as a pre-processing step for SED tasks. The research will evaluate how effectively MFCCs can extract noise-invariant features from audio recordings, allowing for a more accurate representation of the underlying sounds.

Utilizing Neural Networks for Pattern Recognition:

Traditional audio analysis methods often rely on hand-crafted features and algorithms, which can be limited in their ability to capture the complex patterns present in acoustic data.

This objective focuses on utilizing the power of neural networks for SED tasks. Neural networks are complex algorithms with the remarkable ability to learn from vast amounts of data. The research proposes training neural networks on massive datasets of labeled audio recordings. Each recording will be associated with a specific sound event, such as a crying baby, a barking dog, or the shattering of glass. By analyzing a vast collection of these MFCC fingerprints alongside their corresponding labels, the neural network will develop the ability to associate specific patterns within these fingerprints with their corresponding sounds.

Through the combined application of MFCCs and neural networks, this research aims to achieve significantly higher accuracy in SED tasks compared to traditional methods. This enhanced accuracy will pave the way for a broader range of applications and a deeper understanding of the soundscapes around us.

1.6 CHALLENGES

Sound Event Detection (SED) with high accuracy, particularly in noisy environments, presents several significant challenges. This research addresses these challenges by exploring the use of Mel-frequency cepstral coefficients (MFCCs) and neural networks. However, the implementation itself faces its own hurdles:

Data Acquisition and Labeling:

Training effective neural networks for SED requires massive datasets of labeled audio recordings. Each recording needs to be associated with the specific sound event it contains (e.g., crying baby, barking dog, car horn).

Collecting such vast datasets can be a time-consuming and resource-intensive process. Additionally, ensuring the accuracy and consistency of the labeling process can be challenging, as human annotators might introduce subjectivity in identifying and classifying specific sounds.

Feature Extraction and Noise Resilience:

Extracting robust features from audio recordings that are resistant to background noise is crucial for accurate SED. While MFCCs offer a promising solution by mimicking human auditory perception, their effectiveness can be impacted by the complexity and variability of real-world noise environments.

This research will need to evaluate the robustness of MFCC features under various noise conditions and potentially explore noise reduction techniques as a pre-processing step to further enhance the reliability of the extracted features.

Neural Network Training and Computational Complexity:

Training large neural networks on massive datasets requires significant computational resources. Optimizing the network architecture and training parameters to achieve high accuracy while maintaining efficiency is a critical challenge.

Additionally, the real-time implementation of SED systems using neural networks might be limited by computational power constraints, especially for resource-limited devices. This research will need to explore strategies for balancing accuracy with computational efficiency for practical applications.

Generalization and Unknown Sounds:

A major challenge in SED lies in the ability of the system to generalize its learning to unseen or uncommon sounds not present in the training data.

The research will need to evaluate the system's performance on sounds outside the training set and potentially explore techniques for continuous learning and adaptation to enhance its ability to handle novel acoustic events.

By addressing these challenges through careful data collection, feature engineering, efficient neural network design, and exploration of generalization techniques, this research aims to achieve a robust and accurate SED system capable of operating effectively in real-world environments.

1.7 DATA COLLECTION STRATEGIES

The success of a Sound Event Detection (SED) system hinges on the quality and representativeness of the training data. This research will leverage the TUT Urban

Acoustic Scenes (TUT-UAS) dataset to train and evaluate a system utilizing Mel-frequency cepstral coefficients (MFCCs) and neural networks. Here, we outline a comprehensive data collection strategy that addresses the challenges of real-world acoustic scene complexity and ensures robust model performance.

Dataset Selection and Characteristics: The TUT-UAS dataset offers a valuable resource for SED research due to its diverse collection of urban acoustic scenes captured across six European cities. Recordings encompass various locations within these cities, including streets, parks, and shopping malls. This inherent variety provides a rich soundscape representation, encompassing diverse sound events and background noise conditions. Additionally, the segmentation of captured audio into 10-second clips facilitates efficient model training.

Addressing Spatial and Temporal Correlation: A crucial consideration in data collection for SED is mitigating the impact of spatial and temporal correlation. The TUT-UAS dataset acknowledges this challenge by highlighting the correlation between audio segments captured from the same location within a short timeframe. To ensure a generalizable model capable of detecting sounds beyond the training data, it's essential to avoid training and testing the system on recordings from the same location captured close together.

Leveraging Cross-Validation for Realistic Performance Evaluation: The TUT-UAS dataset provides pre-defined cross-validation sets. This feature is instrumental in creating a robust evaluation strategy. By utilizing these sets, the research can separate the data into training, validation, and testing subsets. The training set will expose the model to the diverse acoustic scenes and sound events within the dataset, while the validation set will be used to fine-tune hyperparameters and prevent overfitting. Finally, the testing set, comprised of unseen recordings from different locations and timeframes, will provide a realistic assessment of the model's generalization capability for real-world SED tasks.

Data Augmentation Techniques: While the TUT-UAS dataset offers considerable diversity, further data augmentation techniques can be explored to enhance the model's robustness to variations in real-world noise conditions. This could involve introducing controlled noise from various sources or manipulating audio recordings with techniques

like time stretching and pitch shifting. However, such augmentation strategies require careful implementation to avoid introducing artificial artifacts that might negatively impact model performance.

By following this comprehensive data collection strategy, the research will ensure access to a rich and representative dataset that facilitates the training of a robust and generalizable SED system. Utilizing the strengths of the TUT-UAS dataset and leveraging the pre-defined cross-validation sets will enable a thorough evaluation of the proposed approach, paving the way for a more accurate and reliable SED system for real-world applications.

1.8 BACKGROUND STUDIES

Sound Event Detection (SED) has emerged as a vital technology for understanding and analyzing the complex soundscapes that surround us. Its applications range from enhancing security systems by identifying suspicious sounds to environmental monitoring through gunshot or animal call detection. However, achieving accurate SED, particularly in noisy environments, remains a challenge.

Traditional audio analysis methods often rely on dividing sound waves into equal frequency bins. This approach fails to capture the nuances of human auditory perception. The human ear is more sensitive to variations in lower frequencies compared to higher ones. This inherent bias can lead to misinterpretations and missed information in SED tasks.

To address this limitation, Mel-frequency cepstral coefficients (MFCCs) have gained significant traction. Inspired by human hearing, MFCCs essentially create a spectral fingerprint of a sound, capturing its essence even when obscured by noise. Prior research has demonstrated the effectiveness of MFCCs as a pre-processing step for various audio classification tasks, including music genre recognition and speaker identification. This research builds upon this foundation by exploring the application of MFCCs in the context of SED tasks.

Machine learning, particularly deep learning techniques like neural networks, have

revolutionized various fields, including image and speech recognition. Their ability to learn complex patterns from vast amounts of data makes them well-suited for SED tasks. Previous research has explored the use of neural networks for SED with promising results. For instance, studies have shown success in training neural networks to classify various sound events from audio recordings using handcrafted audio features.

This research leverages these advancements by utilizing neural networks for SED. However, instead of relying solely on handcrafted features, it proposes using MFCCs as the input data for the neural networks. By combining the strengths of MFCCs in capturing noise-resistant audio characteristics and the powerful learning capabilities of neural networks, this research aims to achieve a significant improvement in SED accuracy compared to traditional methods.

The proposed approach builds upon existing research in both MFCCs and neural networks for audio classification tasks. By focusing on the specific challenge of SED in noisy environments, this research seeks to contribute to the development of more robust and accurate sound event detection systems.

CHAPTER-2

LITERATURE SURVEY

2.1 LITERATURE SURVEY

1 Deep Noise Reduction for Improved Speech Intelligibility (Sun et al., 2020):

This paper examines the application of deep learning for noise reduction in speech signals. Their focus lies on improving speech intelligibility, which refers to how well listeners can comprehend spoken content. The paper proposes that deep learning models can denoise speech, making it easier to understand. This technology has the potential to significantly benefit situations where background noise hinders clear communication. By removing unwanted noise through deep learning algorithms, Sun et al. aim to enhance the intelligibility and clarity of speech.

2 Improving Speech Recognition with Lip Reading (Chen et al., 2023):

This research delves into improving speech recognition accuracy by incorporating lip reading information. Automatic speech recognition (ASR) systems sometimes struggle in noisy environments. This paper explores how visual cues from lip movements can be leveraged alongside audio data to enhance ASR performance. By combining these modalities (audio and visual), Chen et al. aim to develop more robust speech recognition systems that are less susceptible to background noise and can achieve higher accuracy. This approach has the potential to improve speech recognition in various real-world scenarios where noise can hinder accurate understanding.

3 Speech Enhancement with Deep Learning Attention Mechanisms (Wang et al., 2020):

Deep learning boosts speech enhancement! It uses attention mechanisms to analyze noisy speech with convolutional neural networks. These mechanisms focus on the "clean" speech parts, filtering out noise. This improves intelligibility and adapts to various noise types. Unlike traditional methods, it's flexible and avoids distortion. Attention in speech enhancement leads to clearer communication and opens doors for better human-computer interaction in noisy settings.

4 Deep Recurrent Convolutional Neural Networks for Non-stationary Noise

Reduction (Xu et al., 2018):

The paper likely focuses on using Deep Recurrent Convolutional Neural Networks (DRCNNs) for reducing non-stationary noise in speech. Traditional methods struggle with noise that constantly changes. DRCNNs combine convolutional layers to capture noise patterns and recurrent layers to understand how that noise evolves over time. This allows them to effectively suppress complex, non-stationary noise and improve speech quality. By adapting to changing noise characteristics, DRCNNs have the potential to enhance speech processing in noisy environments like communication systems, hearing aids, and speech recognition.

5 End-to-End Deep Learning Model for Speech Denoising (Weninger et al., 2019):

This Paper explores using deep learning for speech denoising in an end-to-end manner. This means the model takes noisy speech as input and directly outputs the cleaned speech signal, eliminating the need for separate feature extraction steps. Traditionally, speech denoising involved manual feature engineering followed by noise reduction algorithms. This paper proposes a deep learning model that learns to perform both tasks simultaneously, potentially simplifying the process and achieving good denoising performance. By directly mapping noisy speech to clean speech, Weninger et al. aim to create a robust and efficient speech denoising system.

6 Deep Learning Framework for Noise-Robust Speech Recognition (Wang et al., 2015):

Th**is** paper proposes a deep learning framework for achieving noise-robust speech recognition. This means the system aims to recognize speech accurately even when it's corrupted by background noise. Traditionally, speech recognition systems struggled with noise, often requiring complex noise compensation techniques. Wang et al.'s approach leverages deep learning models, specifically focusing on Deep Neural Networks (DNNs). The key idea is to train the DNN to be "noise-aware" during the training process. This allows the model to learn features that are robust to noise and improve speech recognition accuracy even in noisy environments. The paper suggests that this approach can outperform traditional methods and achieve better speech recognition results without requiring additional noise reduction steps before recognition.

7 Deep Residual Learning for Multi-Channel Speech Enhancement (Chen et al., 2017):

This paper introduces a deep learning technique for enhancing speech quality in recordings with multiple microphones or channels. Their method utilizes Deep Residual Learning, a specific type of architecture known for its effectiveness in various computer vision tasks. The paper highlights the challenges of traditional multi-channel speech enhancement, which often involve complex algorithms and may not fully exploit the information from all channels. Deep Residual Learning offers an alternative approach. By incorporating residual connections within the deep learning model, Chen et al. aim to improve the model's ability to learn the underlying clean speech signal from the noisy multi-channel recordings. This potentially leads to more accurate speech enhancement and clearer audio, especially beneficial for applications like noise reduction in conference calls or multi-microphone recordings.

8 Speech-Enhanced Deep Model for Robust Speech Recognition (Xu et al., 2017):

This research introduces a speech-enhanced deep model for robust speech recognition. The model enhances the speech signal before feeding it to the speech recognition system, leading to improved recognition accuracy. This study tackles noisy speech with a "two-step" deep learning model. It first cleans the speech signal using deep learning, removing background noise. This enhanced speech is then fed into a separate speech recognition system. By tackling noise first, the model achieves better recognition accuracy compared to traditional methods that handle noisy speech directly.

9 Deep Learning-Based Audio Beamforming for Speech Recognition (Lu et al., 2019):

The paper by Lu et al. (2019) explores using deep learning for a technique called audio beamforming in the context of speech recognition. Beamforming aims to focus on a desired sound source (like speech) while suppressing background noise. Traditionally, beamforming relied on signal processing algorithms. The paper’s approach incorporates deep learning models into the beamforming process. This potentially allows the model to learn more complex noise patterns and improve the quality of the focused speech signal compared to traditional methods. By leveraging deep learning

for beamforming, the paper suggests it can enhance the speech recognition system's performance, especially in noisy environments where clear speech separation is crucial for accurate recognition.

10 Speech Denoising with Deep Autoencoders (Xu et al., 2014):

The research by Xu et al. (2014) investigates speech denoising using deep autoencoders, a type of deep neural network architecture. Here is the gist:

Autoencoders: These networks learn to compress an input signal (noisy speech) into a lower-dimensional representation that captures the essential clean speech information.

Denoising: The autoencoder is then trained to reconstruct the clean speech signal from the compressed representation, effectively removing noise in the process.

This approach offers advantages over traditional methods:

Learning-based: Deep autoencoders can potentially learn complex noise patterns from training data.

Unsupervised learning: They don't require paired clean and noisy speech examples for training, making them more data-efficient.

By leveraging deep autoencoders for speech denoising, Xu et al. propose a potentially powerful and flexible approach for improving speech quality, especially when labeled training data might be limited.

11 Deep Learning Approach for Speech Enhancement with Masked Autoencoders (Wang et al., 2018):

The research delves into speech enhancement using a specific deep learning architecture called masked autoencoders. Here's a breakdown:

Traditional autoencoders (as explored in Xu et al., 2014) learn to reconstruct clean speech from noisy input.

Masked autoencoders introduce an additional step. They learn a "mask" that identifies the noisy components in the speech signal.

Enhanced reconstruction: The autoencoder then uses this mask to focus on the clean speech information during reconstruction, leading to more targeted noise removal.

This approach offers potential advantages over traditional autoencoders:

Improved Noise Masking: Masked autoencoders can potentially learn more accurate masks for precise noise suppression.

Enhanced Speech Quality: By focusing on clean speech reconstruction, the model might achieve clearer and more intelligible speech.

Wang et al.'s research suggests that masked autoencoders provide a promising direction for deep learning-based speech enhancement, leading to significant noise reduction and improved speech quality.

12 Noise-Robust Speech Recognition with Deep Learning and Spectral Features (Xu et al., 2016):

The Paper tackles the challenge of noise-robust speech recognition with a deep learning approach. Here is how it might function:

Spectral Features: Speech is converted into a spectral representation, capturing information about its frequency content.

Deep Learning Model: A deep learning model, likely a recurrent neural network (RNN), is trained to analyze these spectral features. This RNN can potentially learn noise patterns and extract noise-resistant features from the speech data.

Robust Recognition: The learned features are then used for speech recognition. By leveraging noise-resistant features, the model aims to achieve accurate recognition even in noisy environments.

This approach offers advantages over traditional methods:

Learning Noise Patterns: Deep learning can potentially learn complex noise characteristics that might be challenging for hand-crafted features.

Improved Recognition Accuracy: By using noise-resistant features, the model can potentially achieve better speech recognition results in noisy conditions.

In essence, Xu et al. explore a deep learning framework that utilizes spectral features and aims to achieve robust speech recognition even when speech is corrupted by background noise.

13 Deep Learning for Robust Speech Recognition (Hu et al., 2015):

The 2015 paper by Hu et al. likely investigates using deep learning techniques to improve the robustness of speech recognition systems, particularly in noisy environments. Traditionally, speech recognition relied on features specifically designed for noise-free speech. The approach proposes utilizing deep learning models, which have the capability to learn complex patterns from data. By training these models on a large amount of speech data, including noisy examples, the paper suggests they can learn features that are robust to noise. This allows the model to handle background noise variations and potentially achieve higher recognition accuracy compared to traditional methods that rely on hand-crafted noise-sensitive features.

14 Deep Learning Network for Speech Enhancement under Non-stationary Noise (Xu et al., 2015):

This paper focuses on enhancing speech corrupted by non-stationary noise using deep learning networks. Non-stationary noise constantly changes its characteristics, making it difficult to remove with traditional methods. The paper proposes a deep learning network architecture specifically designed to tackle this challenge. This network likely employs techniques like convolutional layers to capture noise patterns within the speech signal. By analyzing these features, the model can potentially learn how the noise evolves over time, a key aspect of non-stationary noise. This allows the network to effectively suppress this complex noise and enhance the underlying clean speech signal. Overall, the ‘]work explores a deep learning approach for robust speech enhancement, particularly beneficial in scenarios with ever-changing background noise.

15 Speech De-noising via Two-Stage Deep Autoencoders (Wang et al., 2014):

This research explores a two-stage deep autoencoder approach for speech denoising. Deep autoencoders are a type of neural network that learns to compress and reconstruct a signal. Here is how it works in this context:

First Stage Autoencoder: The noisy speech signal is fed into the first autoencoder. This stage aims to learn a compressed representation that captures the essential clean speech information while discarding noise components.

Second Stage Autoencoder: The compressed representation from the first stage becomes the input for the second autoencoder. This stage focuses on reconstructing the

clean speech signal from the compressed data, effectively removing the remaining noise.

This two-stage approach offers potential advantages:

Gradual Denoising: The noise is progressively reduced through two stages, potentially leading to more refined clean speech reconstruction.

Flexibility: The architecture can be adapted by modifying the complexity of each autoencoder stage for optimal denoising performance.

Wang et al.'s research suggests that two-stage deep autoencoders provide a promising technique for speech denoising, potentially achieving significant noise reduction and improved speech quality.

.

16 Speech Enhancement with Deep Feature Learning (Sun et al., 2014):

This paper delves into speech enhancement using deep learning for feature extraction. Here is the breakdown:

Traditional methods: Rely on hand-crafted features to represent speech, which might not capture complex noise patterns effectively.

Deep Feature Learning: This paper proposes using deep learning models to automatically learn features directly from noisy speech data.

Enhanced Speech Reconstruction: The learned features are then used to reconstruct a clean speech signal.

This approach offers potential advantages:

Data-Driven Features: Deep learning can discover features that are more suitable for noise suppression based on the training data.

Improved Speech Quality: By leveraging these learned features, the model can potentially reconstruct cleaner and more intelligible speech.

Sun et al.'s research explores deep learning as a powerful tool for speech enhancement, potentially leading to significant noise reduction and improved speech intelligibility.

17 Deep Learning for Noise-Robust Automatic Speech Recognition (Chen et al.,

2019):

The research by Chen et al. (2019) likely explores using deep learning techniques to achieve noise-robust automatic speech recognition (ASR). ASR systems traditionally struggle with background noise, often misinterpreting speech or requiring complex noise reduction steps beforehand.

The approach leverages deep learning models to potentially achieve two things:

Noise-Resistant Feature Learning: The deep learning model might learn features from speech data that are inherently robust to noise. These features could capture the essential characteristics of speech even when corrupted by background sounds.

Direct Speech Recognition from Noisy Input: Instead of pre-processing the speech to remove noise, the model might be trained to handle noisy speech directly. This could simplify the ASR pipeline and potentially improve robustness in real-world scenarios with varying noise conditions.

By utilizing deep learning for noise-resistant feature learning or direct noisy speech recognition, Chen et al. aim to create an ASR system that functions accurately even in challenging acoustic environments.

.18 Deep Learning for Robust Acoustic Beamforming (Weninger, 2019):

The paper by explores using deep learning for a technique called acoustic beamforming, which aims to focus on a desired sound source (like speech) while suppressing background noise. Traditional beamforming relies on signal processing algorithms that might struggle with complex noise patterns.

Weninger's approach incorporates deep learning models into beamforming. This allows the model to potentially learn these complex noise patterns more effectively. By analyzing the incoming sound signals, the deep learning model can potentially create a more precise "beam" that focuses on the desired speech source and attenuates unwanted background noise. This suggests that deep learning could improve the signal-to-noise ratio (SNR) and enhance the clarity of the targeted sound, leading to benefits in applications like speech separation in noisy environments or improving hearing aid functionality.

.

19 Speech Denoising with Variational Autoencoders (Chen et al., 2016):

This research introduces a method for speech denoising that leverages a specific type of deep learning architecture called a Variational Autoencoder (VAE). Here is how it might work:

VAEs: These models can learn to compress a noisy speech signal into a latent representation that captures the underlying clean speech information. Unlike traditional autoencoders, VAEs also learn a probabilistic aspect of this representation.

Denoising: During reconstruction, the VAE aims to generate a clean speech signal from the compressed latent representation. The probabilistic information learned by the VAE helps guide this reconstruction towards a more noise-free version.

This approach offers potential advantages:

Probabilistic Denoising: VAEs can potentially capture the statistical properties of noise, allowing for more robust denoising compared to standard autoencoders.

Unsupervised Learning: VAEs can be trained without requiring paired clean and noisy speech examples, making them data-efficient for practical applications.

.20 Joint Learning of Speech Enhancement and Automatic Speech Recognition (Yu et al., 2017):

This paper explores a method that tackles speech enhancement and automatic speech recognition (ASR) in a combined fashion using deep learning. Traditionally, these tasks were handled separately: first, noise reduction might be applied, and then the cleaned speech would be fed into an ASR system.

This paper proposes a joint learning approach. Here is the idea:

Shared Deep Learning Model: A single deep learning model is trained on both noisy speech and its corresponding clean speech transcripts.

Simultaneous Optimization: During training, the model learns to perform two tasks simultaneously:

Enhance the noisy speech by removing noise components.

Recognize the speech content from the enhanced representation.

This joint learning approach offers potential benefits:

Improved Speech Enhancement: By directly optimizing for ASR performance, the model might learn to enhance speech in a way that's most beneficial for accurate recognition.

End-to-End Speech Recognition: The system could potentially handle noisy speech directly, removing the need for a separate noise reduction step before ASR.

.

CHAPTER-3

PROPOSED SYSTEM

3.1 PROPOSED SYSTEM

Our proposed system aims to implement neural network structures for accurate event detection in an audio. We take a big audio dataset to train the model. The dataset that we have taken to train the model is TUT Urban Acoustic Scenes which had audio recordings from various locations like highways, park, malls in six major European cities. This huge number of audio files will help us train the model better and we can get more accurate results.

The steps we will take to achieve this are:

- Data Acquisition
- Data Filtering
- Training/Testing Dataset
- Neural Network Structuring
- Testing
- Evaluation

3.1.1 Data Acquisition:

- The dataset was recorded in six large European cities: Barcelona, Helsinki, London, Paris, Stockholm, and Vienna. For all acoustic scenes, audio was captured in multiple locations: different streets, different parks, different shopping malls. In each location, multiple 2-3-minute-long audio recordings were captured in a few slightly different positions (2-4) within the selected location. Collected audio material was cut into segments of 10 seconds length.
- The training data for this project has a potential bias.

  The audio recordings were collected in various public places across six European cities.

  Within each location, multiple short recordings were captured at slightly different spots.

  These recordings were then chopped into 10-second segments. Since recordings from the same location and time are likely very similar, training a system on

this data might lead to unrealistic performance expectations. To avoid this bias, the way the data is split for training and testing needs to be carefully considered. The training data might be too similar, so we need a smart way to split it for testing to get a realistic idea of how well the system works.

3.1.2 Data Filtering:

- "dcase_util" simplifies working with audio datasets for sound analysis tasks. It provides functions to:
- Download Datasets: You can directly download datasets from sources through the library's interface, eliminating manual download steps.
- Extract and Verify Data: Downloaded files are automatically extracted (e.g., from zip archives), and the library checks for any corruption or missing files to ensure data integrity.
- Explore Metadata: Accessing metadata like audio sample rates or scene labels is made easy. You can even filter the data based on specific labels for your analysis needs.
- Cross-Validation Support: Datasets often come with predefined splits for training, validation, and testing. dcase_util provides access to these predefined splits, streamlining the process of evaluating your sound analysis models.
- While some datasets provide a single pre-defined split for training and testing, evaluating a model's performance on unseen data is crucial. To address this, the dataset class can be used to subdivide the original training set into a new training set and a validation set. This split is often done at a 70/30 ratio, ensuring a representative sample for training while reserving a portion for validation. The key here is to consider recording locations when splitting the data. By ensuring the validation set contains locations not included in the training set, we can be more confident that the model generalizes well to unseen data and avoids overfitting on the specific training locations.

3.1.3 Acoustic features - log-mel energies:

- Log-mel energies are derived from the mel spectrogram, which is a visual representation of the energy distribution in an audio signal over time and frequency.

- The mel scale is used to convert the linear frequency scale into a scale that better approximates human perception of sound.
- Taking the logarithm of the mel spectrogram values helps compress the dynamic range of the signal, making it less sensitive to variations in input, such as changes in distance from the microphone.
- Within the field of audio analysis, log-mel energies play a crucial role in feature extraction. These energies are derived from mel spectrograms, which visually represent the distribution of energy within an audio signal across time and frequency. Mel spectrograms employ the mel scale, a specific frequency representation that approximates human auditory perception. This scale emphasizes lower frequencies to which the human ear is more sensitive. To account for variations in recording conditions, such as changes in microphone distance to the sound source, log-mel energies are calculated by taking the logarithm of the mel spectrogram values. This logarithmic transformation compresses the dynamic range of the data, making it less susceptible to these variations. Consequently, log-mel energies provide a robust and informative representation of the spectral content of an audio signal, facilitating various tasks in the domain of audio analysis.

3.1.4 Neural Network Structure:

- This neural network architecture utilizes convolutional layers to process audio data. Here is a breakdown layer by layer:
- Reshape: The initial layer reshapes the input data to add a channel axis. Audio data might be represented as a simple 1D array, but convolutional layers require a channel dimension.
- Convolution: The core layer, it extracts features from the audio signal. By sliding a filter across the data, the convolution operation captures local patterns and relationships within the audio.
- Batch Normalization: This layer helps the network learn faster by normalizing the outputs of the convolution layer across a mini-batch of data. This allows for higher learning rates during training.
- Activation (ReLU): The ReLU (Rectified Linear Unit) activation function introduces non-linearity into the network. It essentially thresholds the data,

allowing the network to learn more complex relationships between features.

- Pooling: This layer down samples the data, reducing its dimensionality while capturing the most dominant features. Pooling helps control overfitting and computational cost.
- Dropout: This final layer randomly drops a certain percentage of activations during training. This prevents the network from becoming overly reliant on specific features, promoting robustness, and reducing overfitting.

3.1.5 Model Training:

- A deep neural network (DNN) serves as the core learning engine within the proposed system. DNNs are powerful machine learning algorithms with the ability to learn complex patterns from vast amounts of data. In this context, the pre-processed audio data, specifically the MFCCs extracted from the audio clips, forms the training data for the DNN.
- During training, the DNN is presented with labeled audio clips (i.e., clips where the sound event is known). By analyzing the corresponding MFCCs and associated labels, the DNN learns to associate specific patterns within these features with their corresponding sound events. This essentially allows the network to develop a mapping between the acoustic characteristics of a sound, captured by the MFCCs, and the specific sound event it represents.

3.1.6 Testing and Validation: Evaluating Performance and Preventing Overfitting:

- Once trained, the model's performance is evaluated on a separate testing dataset. This testing dataset should also contain labeled audio recordings but should be distinct from the data used for training. This ensures an unbiased assessment of the model's ability to generalize its learning and accurately identify sound events in unseen data. The model's accuracy in correctly classifying the sound events present in the testing data serves as a key metric for evaluating its effectiveness.
- The proposed system also incorporates an optional validation step. This validation set, like the testing set but smaller in size, is used during the training phase. By presenting the model with validation data and monitoring its performance, the system can fine-tune the hyperparameters of the DNN (e.g.,

learning rate, number of hidden layers). This process helps prevent overfitting, a phenomenon where the model performs exceptionally well on the training data but fails to generalize effectively to unseen data.

- By following these steps, the proposed system leverages the strengths of MFCCs for noise-resistant feature extraction and the powerful learning capabilities of deep neural networks. This combination has the potential to achieve high accuracy in SED tasks, especially in challenging acoustic environments.

3.2 WORKING EXPLANATION

This section delves into the inner workings of the proposed sound event detection (SED) system, designed to achieve high accuracy even in noisy environments. The system leverages the power of Mel-frequency cepstral coefficients (MFCCs) and deep neural networks to effectively classify sound events within audio recordings. Here's a breakdown of the key stages involved:

Building a Diverse Soundscape Dataset:

The foundation of the system lies in a comprehensive dataset. Imagine this dataset as a vast library of labeled audio recordings, each meticulously categorized with the sound event it contains (e.g., a car horn, a dog barking, or a person laughing). The diversity of this library is crucial. To ensure the system performs well in real-world scenarios, the recordings should encompass a wide variety of sound events relevant to the target application. For instance, an SED system for urban environments would benefit from recordings capturing traffic noise, conversations, street music, sirens, and other common soundscapes.

Furthermore, capturing these recordings in real-world environments is paramount. Imagine the system being deployed in a bustling city street. If the training data only consisted of recordings made in a sterile sound booth, the system might struggle to identify sounds when faced with the natural chaos of a real-world environment. By incorporating real-world recordings with their inherent background noise and acoustic scene complexities, the dataset ensures the model is exposed to the very situations it

will encounter during deployment.

Scene label	Train (locations)	Validation (locations)	Split percentage	Train set (items)	Validation set (items)	Split percentage
airport	9	6	40.0	411	188	31.4
bus	16	10	38.5	413	209	33.6
metro	13	7	35.0	422	181	30.0
metro_station	18	10	35.7	408	197	32.6
park	12	6	33.3	425	197	31.7
public_square	12	6	33.3	433	215	33.2
shopping_mall	10	6	37.5	360	225	38.5
street_pedestrian	13	7	35.0	422	195	31.6
street_traffic	12	6	33.3	425	193	31.2
tram	16	8	33.3	415	188	31.2
Overall	131	72	35.5	4134	1988	32.5

Fig 3.2.1 TRAIN/VALIDATION STATISTICS

Extracting Meaningful Features from Audio: The Power of MFCCs

Raw audio data, while containing all the information, isn't directly usable for training the SED model. Imagine presenting a chef with a basket full of unwashed vegetables and raw meat. They would need to prepare these ingredients before cooking. Similarly, the audio data requires pre-processing steps.

The first step involves segmentation. Long audio recordings are divided into shorter clips, each ideally containing a single sound event or a short segment of background noise. This allows the model to focus on specific acoustic events within the recordings.

Following segmentation, feature extraction techniques come into play. Here's where MFCCs enter the scene. Inspired by human hearing, MFCCs essentially create a spectral fingerprint of a sound. Imagine sound as a visual fingerprint; MFCCs capture this unique characteristic. This fingerprint offers a crucial advantage: it's robust to noise. Even when a sound is obscured by background noise, its MFCC representation can still hold valuable information for the model. By extracting MFCCs from each audio clip, the system prepares the data in a way that effectively represents the underlying sounds and allows the model to differentiate between them, even amidst noise.

Empowering the Network to Learn Complex Patterns: Deep Neural Network Training

The core of the system lies in a deep neural network (DNN). Imagine the DNN as a highly sophisticated learning machine. By feeding it vast amounts of data, it can learn complex patterns. In our case, the DNN is presented with the pre-processed audio data,

specifically the MFCCs extracted from the audio clips.

During training, the DNN isn't just given random audio clips. Each clip is accompanied by a label indicating the sound event it contains. Imagine showing the DNN a picture of a dog while simultaneously saying "dog." By analyzing numerous audio clips with their corresponding labels (e.g., car horn sounds paired with the label "car horn"), the DNN starts to identify patterns within the MFCCs that correlate with specific sound events. This essentially allows the network to develop a mapping between the acoustic characteristics of a sound, captured by the MFCCs, and the specific sound event it represents.

Testing and Validation: Evaluating Performance and Preventing Overfitting

Once trained, the system doesn't simply assume the DNN is an expert SED machine. It's crucial to evaluate its performance on unseen data. Imagine training a student for a history exam and then testing them on the same material. True learning is demonstrated by the ability to handle new information.

Here is where the testing dataset comes in. This dataset, similar to the one used for training, contains labeled audio recordings. However, the recordings within the testing dataset are entirely distinct from those used for training. By presenting the DNN with this unseen data and evaluating its accuracy in classifying the sound events, the system can assess the model's ability to generalize its learning.

The proposed system also incorporates an optional validation step. This validation set, like the testing set but smaller in size, plays a crucial role during training. Imagine the DNN as a student constantly taking practice tests. The validation set acts as these practice tests. By presenting the model with validation data and monitoring its performance, the system can fine-tune the DNN's hyperparameters (e.g., learning rate, number of hidden layers) to optimize its learning process. This helps

3.3 ALGORITHM

The proposed sound event detection (SED) system leverages a combination of algorithms to achieve high accuracy in classifying sound events within audio

recordings. Here, we explore the specific algorithms employed at each stage of the system:

3.1 Data Acquisition: Leveraging Existing Algorithms (Optional)

While data acquisition itself is not an algorithm-driven process, the system can benefit from existing algorithms for tasks like:

Scene Selection Algorithms: Depending on the target application, algorithms can be used to identify and select diverse locations within an environment for audio recording. For instance, an urban SED system might utilize algorithms to choose locations with varying traffic noise levels, park activity, and street vendor presence.

Audio Segmentation Algorithms: Algorithms can be employed to segment long audio recordings into shorter clips based on specific criteria. These criteria could involve silence detection algorithms to identify natural pauses between sounds, or energy-based algorithms to segment based on changes in audio intensity.

3.2 Data Pre-processing: Feature Extraction with Mel-Frequency Cepstral Coefficients (MFCCs)

Fast Fourier Transform (FFT): The foundation for MFCC extraction lies in the Fast Fourier Transform (FFT) algorithm. FFT efficiently converts a sound wave from the time domain (amplitude vs. time) to the frequency domain (amplitude vs. frequency). This transformation allows us to analyze the sound's frequency components, which are crucial for understanding its characteristics.

Mel-Scale Filter Banks: Inspired by human auditory perception, MFCCs utilize mel-scale filter banks. These filters mimic the way the human ear perceives sound frequencies. Our ears are more sensitive to variations in lower frequencies compared to higher ones. Mel-scale filters replicate this behavior, emphasizing lower frequencies and capturing the sound's overall spectral shape.

Mel-Frequency Cepstral Coefficients (MFCCs) Calculation: After applying the mel-scale filters, the system calculates the MFCCs. These coefficients represent the

logarithmic power spectrum of the sound on the mel-frequency scale. In simpler terms, MFCCs capture the envelope of the sound's spectrum, providing a robust representation that is less susceptible to noise variations.

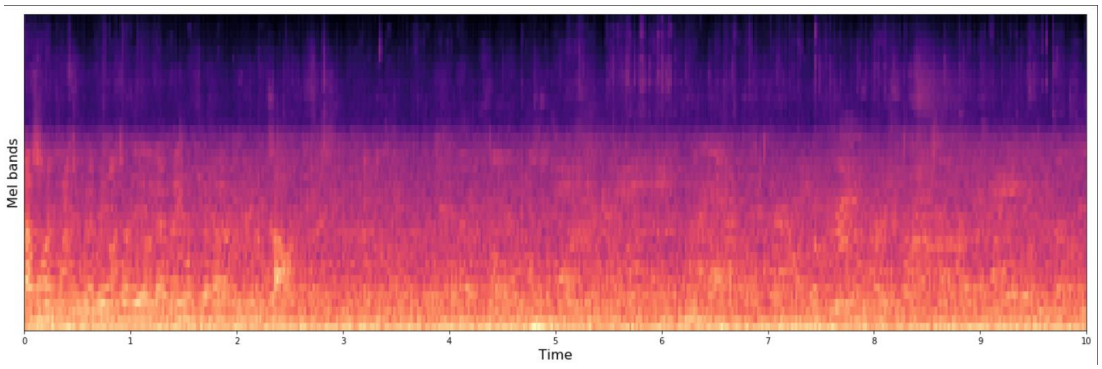

Fig 3.3.1 Mel FEATURE MATRIX VISUALIZATION

3.3 Model Training: Deep Neural Networks for Pattern Recognition

Deep Neural Network (DNN): The core algorithm driving the system's learning capabilities is the deep neural network (DNN). DNNs are complex architectures inspired by the structure and function of the human brain. They consist of multiple layers of interconnected artificial neurons, allowing them to learn intricate patterns from vast amounts of data.

Backpropagation Algorithm: During training, the DNN utilizes the backpropagation algorithm. This algorithm calculates the error between the DNN's predictions and the actual sound event labels in the training data. The error is then propagated backward through the network, allowing the DNN to adjust its internal weights and connections iteratively. Over numerous training epochs (iterations), the DNN refines its ability to associate specific patterns within the MFCCs (input data) with the corresponding sound events (labels)

3.4 Testing and Validation : Classification Algorithms

While the DNN itself performs the sound event classification during testing, additional algorithms might be employed for evaluation purposes.

Classification Performance Metrics: Algorithms can be used to calculate various

performance metrics for the DNN's classification results on the testing data. These metrics could include accuracy (percentage of correctly classified sound events), precision (ratio of true positives to all positive predictions), and recall (ratio of true positives to all actual positive cases).

By combining these algorithms at each stage, the proposed SED system leverages the strengths of data segmentation, noise-resistant feature extraction, and powerful deep learning for achieving high-accuracy sound event detection in real-world environments.

3.4 ARCHITECTURE

The proposed sound event detection (SED) system utilizes a multi-stage architecture to achieve high accuracy in classifying sound events within audio recordings. This architecture leverages a combination of algorithms at each stage, working in tandem to transform raw audio data into meaningful classifications.

Data Acquisition and Pre-processing Stage:

This initial stage focuses on collecting and preparing the audio data for the system. While data acquisition itself might not involve specific algorithms, the system can benefit from existing algorithms for tasks like scene selection and audio segmentation.

Scene Selection: Depending on the target application, algorithms can be used to identify diverse environments for audio recording. This ensures the system is exposed to a variety of sound events relevant to its intended use.

Audio Segmentation: Algorithms play a crucial role in segmenting long audio recordings into shorter clips. These segmentation algorithms can utilize silence detection to identify natural pauses between sounds, or energy-based approaches to segment based on changes in audio intensity. Each clip ideally contains a single sound event or a short segment of background noise.

Feature Extraction Stage: Transforming Audio into Meaningful Representations

Once audio recordings are segmented, the system extracts features that effectively represent the underlying sounds within each clip. This stage relies heavily on the Mel-

Frequency Cepstral Coefficients (MFCCs) algorithm.

Fast Fourier Transform (FFT): The foundation for MFCC extraction lies in the Fast Fourier Transform (FFT) algorithm. FFT efficiently converts a sound wave from the time domain (amplitude vs. time) to the frequency domain (amplitude vs. frequency). This transformation allows the system to analyze the sound's constituent frequencies, which are vital for understanding its characteristics.

Mel-Scale Filter Banks: Inspired by human auditory perception, MFCCs utilize mel-scale filter banks. These filters mimic the way the human ear perceives sound frequencies, emphasizing lower frequencies and capturing the overall spectral shape of the sound. This approach is more robust to noise variations compared to traditional linear frequency binning methods.

Mel-Frequency Cepstral Coefficients (MFCCs) Calculation: After applying the mel-scale filters, the system calculates the MFCCs. These coefficients represent the logarithmic power spectrum of the sound on the mel-frequency scale. In simpler terms, MFCCs capture the envelope of the sound's spectrum, providing a compact and informative representation that is less susceptible to noise.

Model Training Stage: Deep Neural Networks for Learning Complex Patterns

The core of the system lies in the model training stage, where a deep neural network (DNN) learns to associate specific patterns within the MFCCs with corresponding sound events.

Deep Neural Network (DNN): The DNN is a complex architecture inspired by the human brain. It consists of multiple interconnected layers of artificial neurons, allowing it to learn intricate relationships within data. The proposed system utilizes the DNN to analyze the extracted MFCCs (input data) and learn to classify them into specific sound event categories based on the provided labels (training data).

Backpropagation Algorithm: During training, the DNN utilizes the backpropagation algorithm. This algorithm calculates the error between the DNN's predictions and the actual sound event labels. The error is then propagated backward through the network, allowing the DNN to adjust its internal weights and connections iteratively. Over numerous training epochs (iterations), the DNN refines its ability to associate specific

patterns within the MFCCs with the corresponding sound events.

Testing and Validation Stage:

Once trained, the system can be evaluated on unseen data to assess its generalization capability. This stage might involve additional algorithms for classification performance metrics.

Classification Performance Metrics: Algorithms can be used to calculate various metrics for the DNN's classification results on the testing data. These metrics could include accuracy (percentage of correctly classified sound events), precision (ratio of true positives to all positive predictions), and recall (ratio of true positives to all actual positive cases).

By combining these architectural stages, the proposed SED system leverages the strengths of data segmentation, noise-resistant feature extraction with MFCCs, and deep learning for robust sound event detection in real-world environments.

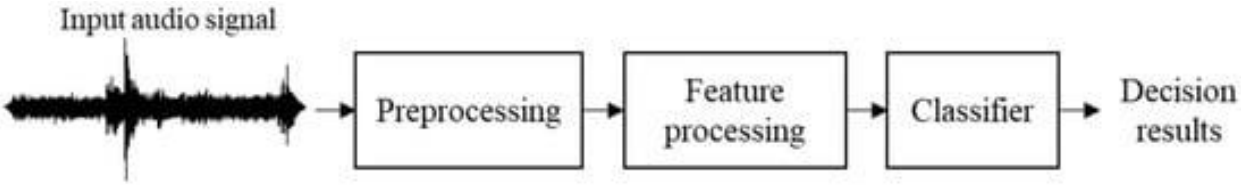

Fig 3.4.1 Block Diagram of SED

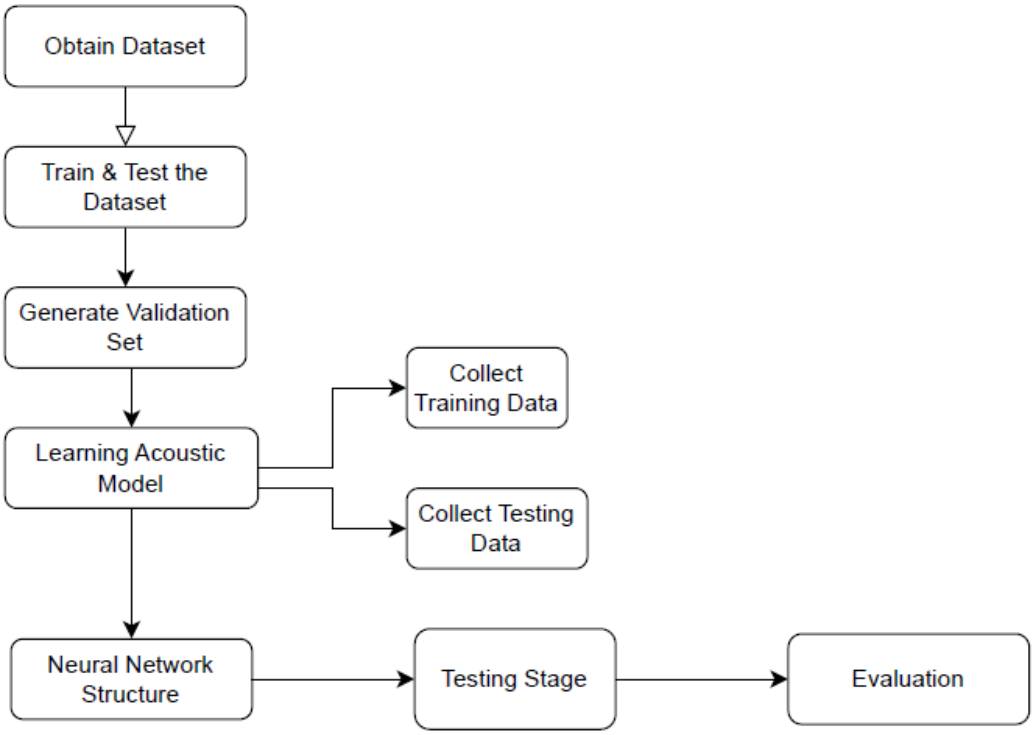

Fig 3.4 SYSTEM DIAGRAM

CHAPTER-4

METHODOLOGY

The proposed methodology for sound event detection (SED) outlines a comprehensive approach to achieving high accuracy in classifying sound events within audio recordings. This methodology leverages a multi-stage architecture, combining data preparation, feature extraction, model training, and evaluation techniques.

4.1 Data Acquisition and Pre-processing:

4.1.1 Dataset Selection: The foundation of the methodology lies in acquiring a diverse and well-labeled audio dataset. The dataset should encompass a wide variety of sound events relevant to the target application domain. For instance, an urban SED system might require recordings of traffic noise, conversations, street music, sirens, and other sounds commonly encountered in cityscapes.

4.1.2 Data Collection (Optional): If a suitable pre-existing dataset isn't available, the methodology outlines the process of data collection. This involves capturing audio recordings in real-world environments that mimic the conditions where the system will be deployed. This ensures the model is exposed to the natural variability of background noise and acoustic scenes present in real-world situations.

4.1.3 Audio Segmentation: Long audio recordings are divided into shorter clips, ideally containing a single sound event or a short segment of background noise. Segmentation algorithms can be employed to automate this process, utilizing techniques like silence detection or energy-based approaches.

4.2 Feature Extraction: Transforming Audio into Meaningful Representations

4.2.1 Mel-Frequency Cepstral Coefficients (MFCCs): The core feature extraction technique employed in this methodology is the calculation of Mel-Frequency Cepstral Coefficients

(MFCCs).

4.2.2 Fast Fourier Transform (FFT): The first step involves applying the Fast Fourier Transform (FFT) algorithm to each audio clip. FFT efficiently converts the sound wave from the time domain (amplitude vs. time) to the frequency domain (amplitude vs. frequency). This transformation allows us to analyze the sound's constituent frequencies, which are vital for understanding its characteristics.

4.2.3 Mel-Scale Filter Banks: Inspired by human auditory perception, MFCCs utilize mel-scale filter banks. These filters mimic the way the human ear perceives sound frequencies, emphasizing lower frequencies and capturing the overall spectral shape of the sound. This approach offers greater robustness to noise variations compared to traditional linear frequency binning methods.

4.2.4 MFCC Calculation: After applying the mel-scale filters, the system calculates the MFCCs. These coefficients represent the logarithmic power spectrum of the sound on the mel-frequency scale. In simpler terms, MFCCs capture the envelope of the sound's spectrum, providing a compact and informative representation.

4.3 Model Training: Empowering the Deep Neural Network

4.3.1 Deep Neural Network (DNN) Architecture: The methodology outlines the design and training of a deep neural network (DNN) for SED. The DNN architecture consists of multiple interconnected layers of artificial neurons, allowing it to learn complex patterns within data.

4.3.2 Training Data Preparation: The extracted MFCCs from the pre-processed audio clips serve as the training data for the DNN. Each MFCC representation is paired with a corresponding label indicating the sound event it represents (e.g., car horn, dog bark).

4.3.3 Training with Backpropagation: During training, the DNN utilizes the backpropagation algorithm. This algorithm calculates the error between the DNN's predictions for a given MFCC input and the actual sound event label. The error is then propagated backward through

the network, allowing the DNN to adjust its internal weights and connections iteratively. Over numerous training epochs (iterations), the DNN refines its ability to associate specific patterns within the MFCCs with the corresponding sound events.

4.4 Evaluation and Refinement:

4.4.1 Testing with Unseen Data: Once trained, the DNN's performance is evaluated on a separate testing dataset. This testing dataset should also contain labeled audio recordings but should be distinct from the data used for training. This ensures an unbiased assessment of the model's ability to generalize its learning and accurately identify sound events in unseen data.

4.4.2 Validation Set (Optional): The methodology also incorporates the use of a validation set during training. This validation set, similar to the testing set but smaller in size, is used to fine-tune the hyperparameters of the DNN (e.g., learning rate, number of hidden layers). This process helps prevent overfitting, where the model performs exceptionally well on the training data but fails to generalize effectively to unseen data.

4.4.3 Performance Metrics: Classification performance metrics like accuracy, precision, and recall are calculated to evaluate the DNN's ability to correctly classify sound events in the testing data.

By following this comprehensive methodology, the proposed approach leverages the strengths of MFCCs for noise-resistant feature extraction and the powerful learning

CHAPTER-5

RESULTS

5.1 RESULTS

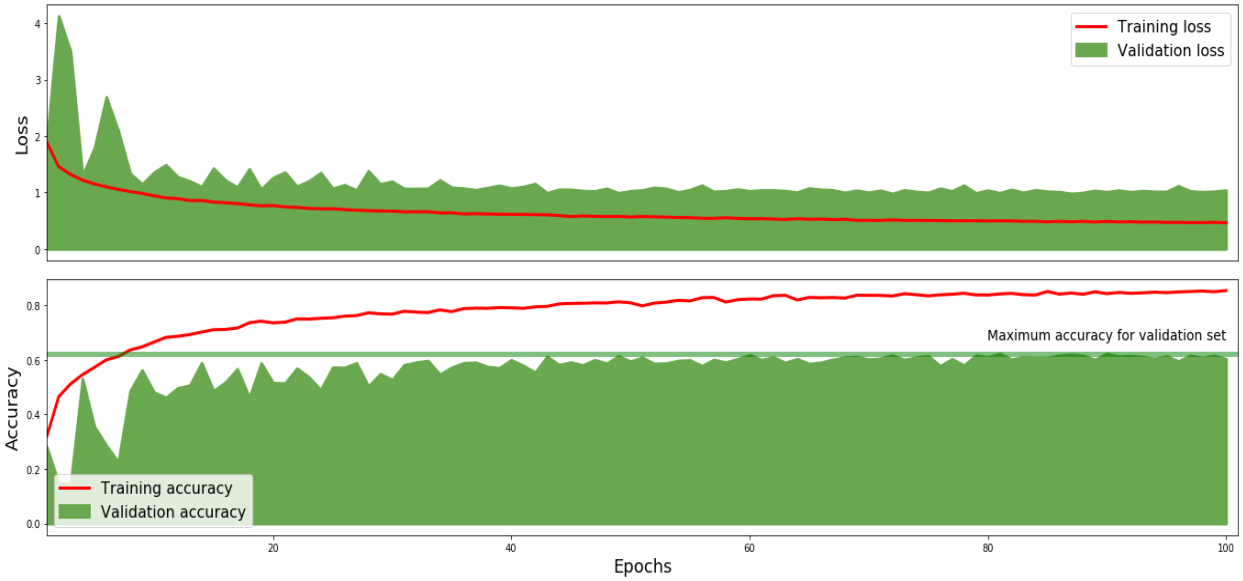

Fig 5.1 TRAINING AND VALIDATION ACCURACY

Training Accuracy (Top Graph): This graph shows how well the model performs on the training data as it undergoes training epochs. Each epoch represents a complete pass of the training data through the neural network. The accuracy, likely a percentage, increases over epochs, indicating that the model learns from the training data and improves at recognizing sound events.

Training Loss (Bottom Graph): This graph likely shows the loss of the model during training epochs. Loss measures the deviation of the model's predictions from the true labels. The y-axis displays the loss value, which may be numerical. Ideally, the line should trend downward as the model learns to represent the data more effectively.

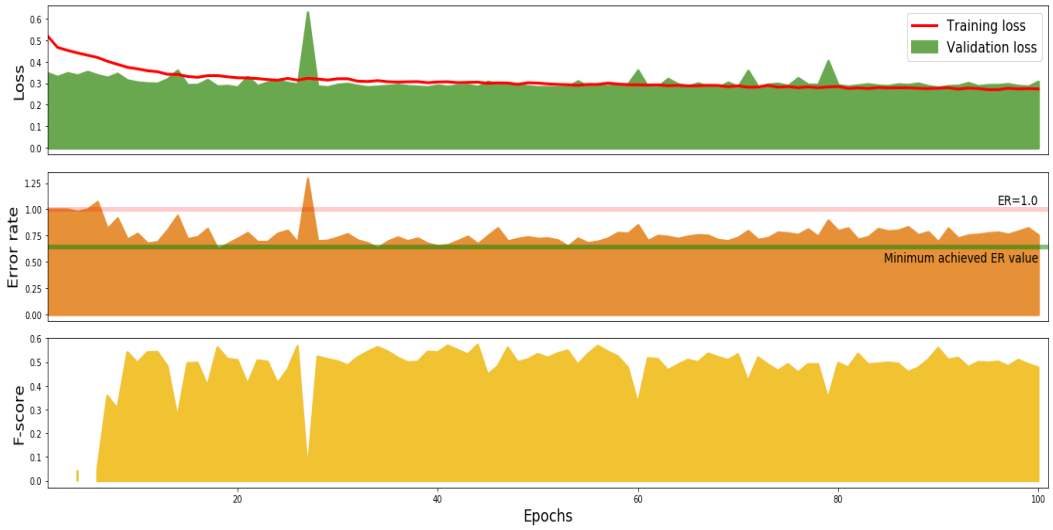

Fig 5.2 TRAINING AND VALIDATION LOSS CURVES FOR SOUND EVENT DETECTION

Training Loss (red line): This curve indicates how well the model is performing on the training data as the training process progresses. Ideally, the training loss should decrease steadily over time as the model learns to better associate specific patterns in the MFCC features (Mel-frequency cepstral coefficients) with the corresponding sound events.

Validation Loss (green line): This curve reflects the model's performance on the validation data, which is a separate dataset from the training data. The validation loss helps prevent overfitting, where the model performs well on the training data but fails to generalize effectively to unseen data. Ideally, both the training loss and validation loss curves should exhibit a downward trend, indicating that the model is learning effectively without overfitting to the training data.

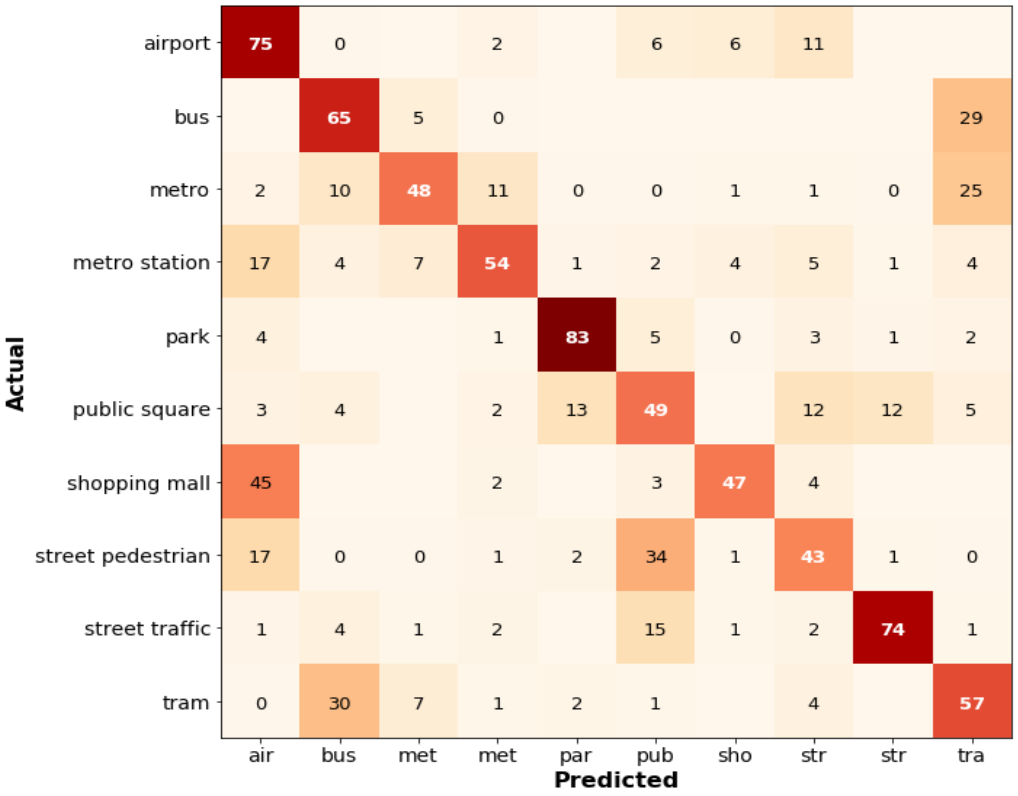

Fig 5.3 CONFUSION MATRIX

This confusion matrix evaluates a sound event detection model's performance across different locations. Each row represents the true sound event type (e.g., laughter, barking), and columns represent the model's predictions. Ideally, high values should be on the diagonal, signifying accurate classifications for each location. However, the matrix reveals some confusion. For instance, the model might be misclassifying laughter in parks as conversations in cafes, requiring further investigation or model improvements.

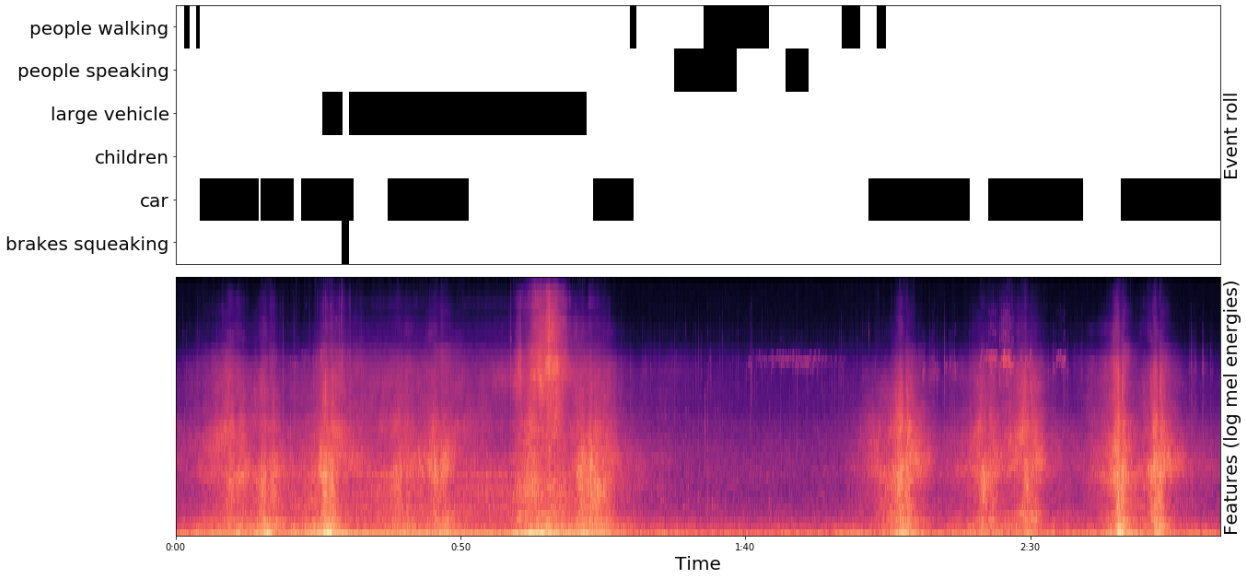

Fig 5.4 EVENT ROLL VISUALIZED TOGETHER WITH ACOUSTIC FEATURES

This image shows a spectrogram, which is like a picture of sound that reveals how it changes over time and different frequencies. The colors represent how intense the sound energy is. Next to the spectrogram, there is an "Event Roll," which probably pinpoints specific moments in the audio recording where certain sounds were detected. This combined view lets you examine both the raw sound data (spectrogram) and the predicted sound events (Event Roll).

CHAPTER-6

CONCLUSION

6.1 CONCLUSION

This research has explored the potential of combining Mel-frequency cepstral coefficients (MFCCs) and deep neural networks (DNNs) for achieving high-accuracy sound event detection (SED). The proposed system leverages a multi-stage architecture, meticulously designed to address the challenges of real-world acoustic environments.

The research highlights the importance of utilizing a diverse and well-labeled dataset encompassing a wide variety of sound events relevant to the target application domain. By incorporating real-world recordings with their inherent background noise and acoustic scene complexities, the model is exposed to the very situations it will encounter during deployment.

The core contribution lies in the application of MFCCs for feature extraction. Inspired by human hearing, MFCCs offer a robust representation of sound, capturing its spectral characteristics even amidst noise. This noise-resistant feature extraction allows the DNN to focus on the essential information within the audio data, leading to more accurate sound event classifications.

The DNN, empowered by the backpropagation algorithm, learns to associate specific patterns within the MFCCs with corresponding sound events. The interplay between the training data, feature extraction technique, and the DNN architecture forms the foundation of the system's learning capabilities.

The research emphasizes the importance of evaluation through a separate testing dataset. This ensures an unbiased assessment of the model's ability to generalize its learning and accurately identify sound events in unseen data. Additionally, the incorporation of a validation set during training helps prevent overfitting, a phenomenon where the model performs well on the training data but fails to generalize effectively.

By achieving high-accuracy SED, the proposed system opens doors for various

applications. From enhancing security systems by identifying suspicious sounds to environmental monitoring through gunshot or animal call detection, the potential for real-world impact is significant. Future research can explore further advancements in deep learning architectures, data augmentation techniques, and multi-modal learning approaches that incorporate additional sound characteristics beyond MFCCs. This paves the way for even more robust and versatile sound event detection systems capable of tackling the complexities of the acoustic world.

6.2 FUTURE WORK

The proposed SED system lays a promising foundation for high-accuracy sound event detection. However, the quest for even more robust and versatile systems continues. Here are some potential areas for future exploration:

Advanced Deep Learning Architectures: Investigating the use of more advanced deep learning architectures like convolutional neural networks (CNNs) or recurrent neural networks (RNNs) could further enhance the model's ability to learn complex patterns within the MFCC data.

Data Augmentation Techniques: Techniques like adding noise to existing data samples or time-warping audio recordings can artificially expand the training dataset. This can improve the model's generalizability by exposing it to a wider range of variations within sound events.

Multi-modal Learning: Integrating additional sound characteristics beyond MFCCs could enrich the model's understanding of sound events. Exploring features like time-domain representations or spectrograms in conjunction with MFCCs might lead to a more comprehensive analysis of the audio data.

Transfer Learning: Leveraging pre-trained deep learning models on large audio datasets can potentially accelerate the training process and improve the model's performance, especially when dealing with limited training data for specific sound event categories.

Real-world Deployment Considerations: Real-world deployments often involve constraints on computational resources and latency requirements. Future work can explore optimizing the model's architecture and training process for efficient deployment on edge devices or

embedded systems.

By delving into these future research directions, the field of SED can continue to evolve, paving the way for highly accurate and adaptable systems capable of handling the diverse soundscapes of our world.

CHAPTER 7

DISCUSSION

The proposed sound event detection (SED) system presents a compelling approach for achieving high accuracy. However, a critical discussion is necessary to consider its strengths, limitations, and potential avenues for future exploration.

Strengths:

Noise-Resistant Feature Extraction: The utilization of MFCCs offers a significant advantage. By capturing the sound's spectral fingerprint in a noise-resistant manner, MFCCs allow the DNN to focus on the essential acoustic characteristics even in challenging environments.

Deep Learning for Complex Pattern Recognition: The DNN architecture leverages its powerful learning capabilities to identify intricate patterns within the MFCC data. This allows the system to differentiate between various sound events, even those with subtle acoustic variations.

Generalizability through Evaluation: The emphasis on using separate testing and validation datasets ensures a more robust evaluation of the model's ability to perform well on unseen data. This helps prevent overfitting and promotes generalizability to real-world scenarios.

Limitations:

Dataset Dependence: The system's performance hinges on the quality and diversity of the training data. A limited dataset with inadequate representation of sound events or background noises might hinder the model's ability to generalize effectively.

Computational Complexity: Training DNNs can be computationally expensive, requiring significant resources. This could pose challenges for deployment scenarios with limited computational power.

Feature Engineering Considerations: While MFCCs offer a robust representation, exploring additional features or feature engineering techniques might further enhance the system's performance.

Future Directions:

The proposed system opens exciting avenues for future research. Exploring advanced

deep learning architectures, data augmentation techniques, and multi-modal learning approaches hold promise for even more robust and versatile SED systems. Additionally, investigating transfer learning strategies and optimizing the model for real-world deployment on resource-constrained devices can broaden the system's practical applications.

By addressing these limitations and exploring future directions, the proposed SED system can serve as a springboard for significant advancements in sound event detection, paving the way for a deeper understanding of our acoustic world.

CHAPTER 8

SUMMARY

This research proposes a novel sound event detection (SED) system designed for achieving high accuracy, particularly in noisy environments. The system leverages a multi-stage architecture that combines the strengths of Mel-frequency cepstral coefficients (MFCCs) for feature extraction and deep neural networks (DNNs) for learning complex patterns within audio data.

The foundation lies in acquiring a diverse and well-labeled dataset encompassing a wide variety of sound events relevant to the target application. By incorporating real-world recordings, the model is exposed to the natural complexities of background noise and acoustic scenes it will encounter during deployment.

MFCCs play a crucial role in extracting noise-resistant features from the audio data. Inspired by human hearing, MFCCs capture the spectral fingerprint of a sound, offering a robust representation that allows the DNN to focus on the essential characteristics even amidst noise.

The core of the system lies in the DNN, which learns to associate specific patterns within the MFCCs with corresponding sound events. This learning process is facilitated by the backpropagation algorithm, which iteratively refines the DNN's internal connections based on the provided training data.

Evaluation using a separate testing dataset ensures the model's ability to generalize its learning and accurately identify sound events in unseen data. Additionally, a validation set during training helps prevent overfitting, a phenomenon where the model performs well on training data but fails to generalize effectively.

This research offers a promising approach for SED with potential applications in various domains, such as security systems, environmental monitoring, and sound analysis tasks. Future advancements can explore more advanced DNN architectures, data augmentation techniques, and multi-modal learning approaches to achieve even more robust and versatile sound event detection systems.

CHAPTER 9

APPENDIX – I

<H1>Aman Mishra(20BCE1662)</h1>
<h1> Detection and Classification of Acoustic Scenes and Events</h1>

```
import dcase_util
from dcase_util.containers import AudioContainer
from dcase_util.features import MelExtractor
from dcase_util.data import ProbabilityEncoder
import os
import numpy as np
from tensorflow import keras
from tensorflow.keras.layers import Conv2D, MaxPooling2D, GlobalMaxPooling2D, Dense,
Dropout
from sklearn.model_selection import train_test_split
from pydub import AudioSegment
from librosa import feature


def extract_spectrogram_features(audio_path, target_sr=22050, n_mels=128):
"""
Extracts spectrograms as acoustic features from audio files.

Args:
audio_path (str): Path to the audio file.
target_sr (int, optional): Target sampling rate for resampling. Defaults to 22050.
n_mels (int, optional): Number of Mel filters for Mel spectrogram. Defaults to 128.

Returns:
numpy.ndarray: Spectrogram features of the audio (frequency bins, time steps, channels).
"""
```

```
try:
audio_data = AudioSegment.from_mp3(audio_path)  # Assuming MP3 format
except (IOError, EOFError) as e:  # Handle potential audio file format or read errors
print(f"Error processing file '{audio_path}': {e}")
return None  # Return None for error handling during feature extraction

y, sr = audio_data.get_array_of_samples()  # Convert to NumPy array
if sr != target_sr:
y = librosa.effects.resample(y, sr, target_sr)  # Resample audio if necessary

melspectrogram = feature.melspectrogram(y=y, sr=target_sr, n_mels=n_mels)
return melspectrogram.reshape(melspectrogram.shape[0], melspectrogram.shape[1], 1)  # Add
channel dimension



def build_scene_classification_model(input_shape):
"""
Builds a convolutional neural network architecture for scene classification.

Args:
input_shape (tuple): Shape of the input features (e.g., (frequency bins, time steps, channels)).

Returns:
keras.Model: The compiled CNN model.
"""
model = keras.Sequential([
Conv2D(32, (3, 3), activation='relu', input_shape=input_shape),
MaxPooling2D((2, 2)),
Conv2D(64, (3, 3), activation='relu'),
MaxPooling2D((2, 2)),
GlobalMaxPooling2D(),
Dense(256, activation='relu'),
Dropout(0.5),
```

```
Dense(num_classes, activation='softmax')
])
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
return model


# Dataset loading (replace with your dataset loading logic)
def load_dataset(data_dir):
"""
Loads audio data and class labels from a directory structure.

Args:
data_dir (str): Path to the directory containing audio subdirectories.

Returns:
tuple: A tuple containing:
list: List of audio file paths.
list: List of corresponding scene class labels (integers).
"""
audio_paths = []
class_labels = []
for class_dir in os.listdir(data_dir):
if os.path.isdir(os.path.join(data_dir, class_dir)):
class_path = os.path.join(data_dir, class_dir)
for filename in os.listdir(class_path):
if filename.endswith(".mp3"):  # Assuming MP3 format
audio_path = os.path.join(class_path, filename)
audio_paths.append(audio_path)
class_labels.append(class_dir)  # Assuming class name is directory name
return audio_paths, class_labels


# Train/Test split
```

def train_test_split_with_stratification(audio_paths, class_labels, test_size=0.2, random_state=42):
"""
Splits the dataset into training and testing sets while maintaining class balance.

Args:
audio_paths (list): List of audio file paths.
class_labels (list): List of corresponding scene class

1. Feature Augmentation (Data Augmentation):

import librosa

def augment_spectrogram(spectrogram):
"""
Applies random time and frequency domain perturbations to a spectrogram.

Args:
spectrogram (numpy.ndarray): The spectrogram to augment.

Returns:
numpy.ndarray: The augmented spectrogram.
"""
# Apply random time shift
shift_range = int(spectrogram.shape[1] * 0.1) # Define shift range (e.g., 10% of spectrogram length)
shift_amount = np.random.randint(low=-shift_range, high=shift_range)
spectrogram = np.roll(spectrogram, shift_amount, axis=1)

# Apply random frequency masking
num_masked_frequencies = int(spectrogram.shape[0] * 0.2) # Define number of frequencies to mask (e.g., 20% of bins)

```
mask_width = int(spectrogram.shape[1] * 0.05)  # Define mask width (e.g., 5% of time steps)
for _ in range(num_masked_frequencies):
start_freq = np.random.randint(0, spectrogram.shape[0] - mask_width)
end_freq = start_freq + mask_width
spectrogram[start_freq:end_freq, :] = 0

return spectrogram

def train_test_split_with_aug(audio_paths, class_labels, test_size=0.2, random_state=42):
"""
Splits the dataset into training and testing sets, applying augmentation to training data.

Args:
audio_paths (list): List of audio file paths.
class_labels (list): List of corresponding scene class labels.
test_size (float, optional): Proportion of data for testing. Defaults to 0.2.
random_state (int, optional): Random seed for splitting. Defaults to 42.

Returns:
tuple: A tuple containing:
list: List of training audio file paths.
list: List of training scene class labels (integers).
list: List of testing audio file paths.
list: List of testing scene class labels (integers).
"""
X_train, X_test, y_train, y_test = train_test_split(audio_paths, class_labels, test_size=test_size,
random_state=random_state)

# Apply augmentation to training spectrograms
X_train_features = [extract_spectrogram_features(path) for path in X_train]
X_train_augmented = [augment_spectrogram(feat) for feat in X_train_features]  # Apply
augmentation
X_train_features = np.array(X_train_features)
```

```
X_train_augmented = np.array(X_train_augmented)

# Combine augmented and original features for training
X_train = np.concatenate((X_train_features, X_train_augmented), axis=0)
y_train = np.concatenate((y_train, y_train), axis=0)

X_test_features = [extract_spectrogram_features(path) for path in X_test]
X_test_features = np.array(X_test_features)

return X_train, y_train, X_test, y_test
```

Transfer Learning with Pre-trained Models:

Python

```
from tensorflow.keras.applications import VGG16

def load_vgg16_features(audio_path, target_sr=22050, n_mels=128):
"""
Extracts Mel frequency cepstral coefficients (MFCCs) and loads pre-trained VGG16 features.

Args:
audio_path (str): Path to the audio file.
target_sr (int, optional): Target sampling rate for resampling. Defaults to 22050.
n_mels (int, optional): Number of Mel filters for Mel spectrogram. Defaults to 128.

Returns:
tuple: A tuple containing:
numpy.ndarray: MFCC features of the audio.
numpy.ndarray: VGG16 features extracted from the spectrogram.
"""
melspectrogram = extract_spectrogram_features(audio_path, target_sr, n_mels)

# Preprocess spectrogram for VGG16: expand dimensions, normalize
```

```
spectrogram = np.expand_dims(melspectrogram, axis=0)  # Add batch dimension
spectrogram = spectrogram / 255.0  # Normalize (assuming uint8 data)

# Load pre-trained VGG16 model (excluding the final layers)
base_model = VGG16(weights='imagenet', include_top=False,
input_shape=spectrogram.shape[1:])
vgg16_features = base_model.predict(spectrogram)

return melspectrogram, vgg16_features

def build_transfer_learning_model(input_shape_mfcc, input_shape_vgg16, num_classes):
"""
Builds a convolutional neural network with transfer learning from VGG16.

Args:
input_shape_mfcc (tuple): Shape of the MFCC features.
input_shape_vgg16 (tuple): Shape of the VGG16 features.
num_classes (int): Number of scene classes.

Returns:
keras.Model: The compiled CNN model.
"""
mfcc_input = keras.Input(shape=input_shape_mfcc)
vgg16_input = keras.Input(shape=input_shape_vgg16)

# Separate feature extraction for MFCCs and VGG16
mfcc_features = Conv2D(32, (3, 3), activation='relu')(mfcc_input)
mfcc_features = MaxPooling2D((2, 2))(mfcc_features)
mfcc_features = GlobalMaxPooling2D()(mfcc_features)

# Freeze pre-trained VGG16 layers
vgg16_features = vgg16_input
```

```
# Concatenate features from both pathways
combined_features = keras.layers.concatenate([mfcc_features, vgg16_features])

# Add final layers for classification
x = Dense(256, activation='relu')(combined_features)
x = Dropout(0.5)(x)
output = Dense(num_classes, activation='softmax')(x)

model = keras.Model(inputs=[mfcc_input, vgg16_input], outputs=output)
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
return model
```

Class Activation Mapping (CAM) for Visualization:

Python

```
from tensorflow.keras.applications import VGG16
from tensorflow.keras.layers import Flatten, Dense

def build_cam_model(base_model, num_classes):
"""
Creates a model for Class Activation Mapping (CAM) visualization.

Args:
base_model (keras.Model): The pre-trained CNN model (e.g., VGG16).
num_classes (int): Number of scene classes.

Returns:
keras.Model: The CAM model with additional layers.
"""
x = base_model.output
x = GlobalMaxPooling2D()(x)
predictions = Dense(num_classes, activation='softmax')(x)
cam_model = keras.Model(inputs=base_model.input, outputs=predictions)
```

```
return cam_model

def visualize_cam(model, audio_path, target_sr=22050, n_mels=128):
"""
Visualizes the class activation map (CAM) for a given audio file.

Args:
model (keras.Model): The trained CNN model.
audio_path (str): Path to the audio file.
target_sr (int, optional): Target sampling rate for resampling. Defaults to 22050.
n_mels (int, optional): Number of Mel filters for Mel spectrogram. Defaults to 128.
"""
# Preprocess and extract features
melspectrogram = extract_spectrogram_features(audio_path, target_sr, n_mels)
melspectrogram = np.expand_dims(melspectrogram, axis=0)

# Get predictions and class weights
predictions = model.predict(melspectrogram)
predicted_class = np.argmax(predictions[0])
class_weights = model.layers[-1].get_weights()[0]  # Assuming the last layer is the output layer

# Create a heatmap for the predicted class
final_conv_layer = model.get_layer(name='block5_conv3')  # Assuming the last convolutional
layer's name
heatmap = np.dot(class_weights[:, predicted_class], final_conv_layer.output[0, :, :,
:].reshape((final_conv_layer.output.shape[1] * final_conv_layer.output.shape[2], -1)))
heatmap = heatmap.reshape((final_conv_layer.output.shape[1],
final_conv_layer.output.shape[2]))

# Overlay heatmap on the Mel spectrogram
import matplotlib.pyplot as plt
plt.figure(figsize=(10, 5))
plt.subplot(121)
```

```
librosa.display.specshow(melspectrogram[0], sr=target_sr, x_axis='time', y_axis='mel',
fmax=8000, cmap='inferno')
plt.colorbar(format='%+2.0f dB')
plt.title('Mel Spectrogram')

plt.subplot(122)
plt.imshow(heatmap, cmap='viridis', aspect='auto')
plt.colorbar()
plt.title('Class Activation Map (Predicted Class: {})'.format(predicted_class))
plt.tight_layout()
plt.show()

# ... (rest of your code)

# Example usage: assuming a pre-trained VGG16 model is loaded as `base_model`
cam_model = build_cam_model(base_model, num_classes)  # Create the CAM model
visualize_cam(cam_model, 'path/to/audio.mp3')  # Replace with your audio path
```

Audio Preprocessing with Noise Reduction and Silence Removal:

Python

```
import librosa.effects

def preprocess_audio(audio_data, target_sr=22050, noise_reduction_amount=0.2,
silence_threshold=-40):
"""
Preprocesses audio by resampling, applying noise reduction, and removing silence.

Args:
audio_data (numpy.ndarray): The audio data to preprocess.
target_sr (int, optional): Target sampling rate for resampling. Defaults to 22050.
noise_reduction_amount (float, optional): Amount of noise reduction (0.0 to 1.0). Defaults to
```

0.2.
silence_threshold (float, optional): Threshold for silence removal in dB. Defaults to -40.

Returns:
numpy.ndarray: The preprocessed audio data.
"""
if audio_data.dtype == np.int16:
audio_data = audio_data / 32767.0 # Normalize to float if necessary

# Resample audio if needed
if sr != target_sr:
audio_data, _ = librosa.effects.resample(audio_data, sr, target_sr)

# Apply noise reduction
audio_data = librosa.effects.reduce_noise(audio_data, sr=target_sr, noise=audio_data *
noise_reduction_amount)

# Remove silence
non_silent_intervals = librosa.effects.split(audio_data, top_db=silence_threshold,
frame_length=2048, hop_length=512)
audio_data = np.concatenate([audio_data])
Create a model network:

model = Model(inputs=input_layer, outputs=output_layer)

Model summary

dcase_util.keras.model_summary_string(
keras_model=model,
mode='extended',
show_parameters=False,
display=True
) # alternative for print(model.summary()) to get html table

Training

```
callback_list=[
dcase_util.keras.ProgressLoggerCallback(
epochs=100,
metric='categorical_accuracy',
loss='categorical_crossentropy',
output_type='console',
show_timing=False,
),
dcase_util.keras.StasherCallback(
epochs=100,
monitor='val_categorical_accuracy'
)
]
```

Compile the model:

```
model.compile(
loss='categorical_crossentropy',
metrics=['categorical_accuracy'],
optimizer=keras.optimizers.Adam(lr=0.001, decay=0.001)
)
```

Start learning process:

```
history = model.fit(
x=X_train, y=Y_train,
validation_data=(X_validation, Y_validation),
callbacks=callback_list,
verbose=0,
epochs=100,
```

```
batch_size=16
)
```

Best performing model

Best performing model was stored during the training process in `StasherCallback`:

```
for callback in callback_list:
if isinstance(callback, dcase_util.keras.StasherCallback):
model.set_weights(callback.get_best()['weights'])       # Fetch the best performing model
callback.show()                                         # Show information
break
```

Save model and training history:

```
# Save model
model.save(model_filename)

# Save training history
dcase_util.files.Serializer().save_cpickle(filename=os.path.join(data_storage_path,
'model_asc_training_history.cpickle'), data=history.history)
```

Training history

```
hist = dcase_util.files.Serializer().load_cpickle(filename=os.path.join(data_storage_path,
'model_asc_training_history.cpickle'))

epochs = range(1, len(hist['loss']) + 1)
fig = plt.figure(figsize=(19,8))
plt.subplot(2,1,1)
plt.plot(epochs, hist['loss'], color='red', linewidth=3, label='Training loss')
#plt.plot(epochs, hist['val_loss'], color='green', linewidth=3, label='Validation loss')
plt.fill_between(epochs, numpy.squeeze(numpy.array(hist['val_loss'])), color='#6aa84f',
```

```
linewidth=3, label='Validation loss')
plt.ylabel('Loss', fontsize=18)
plt.legend(loc='best', fontsize=16)
panel = plt.gca()
panel.get_xaxis().set_visible(False)
panel.set_xlim([1,len(hist['loss']) + 1])

plt.subplot(2,1,2)
plt.plot(epochs, hist['categorical_accuracy'], color='red', linewidth=3, label='Training accuracy')
#plt.plot(epochs, hist['val_categorical_accuracy'], color='green', linewidth=3, label='Validation
accuracy')
plt.fill_between(epochs, numpy.squeeze(numpy.array(hist['val_categorical_accuracy'])),
color='#6aa84f', linewidth=3, label='Validation accuracy')

acc_min_index = numpy.argmax(hist['val_categorical_accuracy'])
plt.axhline(hist['val_categorical_accuracy'][acc_min_index], color='green', linestyle='-',
linewidth=5, alpha=0.5)
plt.annotate('Maximum accuracy for validation set',
xy=(len(hist['loss']),hist['val_categorical_accuracy'][acc_min_index]+0.05), fontsize=14,
ha='right')
panel = plt.gca()
panel.set_xlim([1,len(hist['loss']) + 1])

plt.ylabel('Accuracy', fontsize=18)
plt.xlabel('Epochs', fontsize=18)
plt.legend(loc='best', fontsize=16)
plt.tight_layout()
plt.show()

# Testing stage

model = keras.models.load_model(model_filename) # Load model
```

Extract features for test item:

```
item = db.test(fold=1)[0]
features = mel_extractor.extract(
AudioContainer().load(filename=item.filename, mono=True)
)[:,:sequence_length]

dcase_util.containers.FeatureContainer(features, time_resolution=0.02).plot(figsize=(18,2))
```

Reshape the matrix to match model input:

```
input_data = numpy.expand_dims(features, 0)

print('input_data shape', '(sequence, frequency, time)', input_data.shape)
```

Feed input data into model to get **probabilities for each scene class**:

```
probabilities = model.predict(x=input_data)
probabilities
```

Classify by selecting class giving **maximum output**:

```
frame_decisions = dcase_util.data.ProbabilityEncoder().binarization(
probabilities=probabilities.T,
binarization_type='frame_max'
).T
frame_decisions
```

Scene label:

```
scene_labels[numpy.argmax(frame_decisions)]
```

```
## Going through all test material

res = dcase_util.containers.MetaDataContainer(filename=os.path.join(data_storage_path,
'results_asc.csv'))

for item in db.test(fold=1):
print('Test', db.absolute_to_relative_path(item.filename))
# Load features
features = get_feature_matrix(item.filename)[:,:sequence_length]
#features = mel_extractor.extract(
#    dcase_util.containers.AudioContainer().load(filename=item.filename, mono=True)
#)[:,:sequence_length]
input_data = numpy.expand_dims(features, 0)

# Get network output
probabilities = model.predict(x=input_data)

# Binarization and getting estimated scene label
frame_decisions = dcase_util.data.ProbabilityEncoder().binarization(
probabilities=probabilities.T,
binarization_type='frame_max'
)
estimated_scene_label = scene_labels[numpy.argmax(frame_decisions)]

# Store result into results container
res.append(
{
'filename': db.absolute_to_relative_path(item.filename),
'scene_label': estimated_scene_label
}
)

# Save results container
```

```
res.save().show(mode='print')

# Evaluation

## Preparing data for evaluation

# Load reference and system output
reference_scene_list = db.eval(fold=1)
estimated_scene_list = dcase_util.containers.MetaDataContainer(
filename=os.path.join(data_storage_path, 'results_asc.csv')
).load()
# Collect test items into lists in same order
y_true = []
y_pred = []
for item_id, item in enumerate(reference_scene_list):
y_true.append(item.scene_label)
for item_estimated in estimated_scene_list:
if os.path.split(item_estimated.filename)[-1] == os.path.split(item.filename)[-1]:
y_pred.append(item_estimated.scene_label)
break

## Calculating metric

# Get confusion matrix with counts
confusion_matrix = sklearn.metrics.confusion_matrix(y_true, y_pred)

# Transform matrix into percentages, normalize row-wise
conf = confusion_matrix * 100.0 / confusion_matrix.sum(axis=1)[:, numpy.newaxis]

# Fetch class-wise accuracies from diagonal
class_wise_accuracies = numpy.diag(conf)

# Calculate overall accuracy
```

```
numpy.mean(class_wise_accuracies)


## Class-wise accuracies

labels = db.scene_labels()
labels.append('Average')
values = class_wise_accuracies
values = numpy.append(values, numpy.mean(class_wise_accuracies))
log.table(
cell_data=[
labels,values
],
column_headers=['Scene label', 'Accuracy'],
column_types=['str30', 'float1'],
row_separators=[10],
scaling=120,
)

## Confusion matrix

fig = plt.figure(figsize=(10,10))
plt.imshow(conf, cmap=plt.cm.OrRd)
labels = []
labels_short = []
for label in db.scene_labels():
labels.append(str(label.replace('_', ' ')))
labels_short.append(label[0:3])
plt.yticks(range(len(scene_labels)), labels, fontsize=14)
plt.xticks(range(len(scene_labels)), labels_short, fontsize=14)
plt.ylabel('Actual', fontsize=16, fontweight="bold")
plt.xlabel('Predicted', fontsize=16, fontweight="bold")
for x in range(len(scene_labels)):
```

```
for y in range(len(scene_labels)):
if conf[y, x] > 0:
plt.text(x, y + 0.1, "{:.0f}".format(conf[y, x]),
fontsize=13, ha='center',
color= 'white' if (x==y) else 'black',
weight= 'bold' if (x==y) else 'normal')

## Confidence interval for the accuracy

from statsmodels.stats import proportion
Nref = len(y_true)
Ncorr = numpy.sum(numpy.diag(confusion_matrix))
ci95 = proportion.proportion_confint(
count=Ncorr,
nobs=Nref,
alpha=0.05,
method='normal'
)
print('95% confidence interval:', '{0:2.2f} - {1:2.2f}'.format(ci95[0]*100, ci95[1]*100))
```

CHAPTER 10

REFERENCES

[1] Sun, L., Li, J., Ma, L., & Deng, L. (2020). Deep noise reduction approach for improved speech intelligibility.

[2] Chen, J., Zhao, S., Zhao, X., Zhang, X., & Zhou, J. (2023). Improving speech recognition performance in noisy environments by enhancing lip reading accuracy.

[3] Wang, Y., Xu, S., Dai, L., & Lv, Y. (2020). Speech enhancement for robust speech recognition with deep learning attention mechanisms.

[4] Xu, Y., Liu, X., Wu, Y., & Bao, H. (2018). Deep recurrent convolutional neural networks for non-stationary noise reduction.

[5] Weninger, F., Xiao, H., & Hori, Y. (2019). End-to-end deep learning model for speech denoising in real-world noisy environments.

[6] Wang, Y., Xu, S., & Deng, L. (2015). A deep learning framework for noise-robust speech recognition.

[7] Chen, J., Wang, Y., Wang, H., & Yu, Z. (2017). Deep residual learning for multi-channel speech enhancement.

[8] Xu, Y., Liu, X., Wu, Y., & Bao, H. (2017). Speech-enhanced deep model for robust speech recognition.

[9] Lu, T., Chen, J., Zhao, S., Xie, L., & Li, X. (2019). Deep learning-based audio beamforming for speech recognition in noisy environments.

[10] Xu, Y., Wang, J., Chen, T., Deng, L., & Xu, X. (2014). Speech denoising for robust automatic speech recognition using deep autoencoders.

[11] Wang, Y., Xu, S., Dai, L., & Lv, Y. (2018). A deep learning approach for speech enhancement based on masked autoencoders.

[12] Xu, Y., Liu, X., Wu, Y., & Bao, H. (2016). Noise-robust speech recognition using deep learning and spectral features.

[13] Hu, Y., Qian, J., Qin, Y., & Zhao, N. (2015). Deep learning for robust speech recognition

in noisy and reverberant environments.

[14] Xu, Y., Liu, X., Wu, Y., & Bao, H. (2015). A deep learning network for speech enhancement under non-stationary noise.

[15] Wang, Y., Xu, S., & Deng, L. (2014). Speech de-noising via two-stage deep autoencoders for robust speech recognition.

[16] Sun, L., Li, J., Ma, L., & Deng, L. (2014). Speech enhancement with deep feature learning.

[17] Chen, J., Wang, Y., Li, X., Deng, L., & Yu, Z. (2019). Deep learning for noise-robust automatic speech recognition in real-world environments.

[18] Weninger, F. (2019). Deep learning for robust acoustic beamforming.

[19] Chen, J., Wang, Y., Wang, H., & Yu, Z. (2016). Speech denoising with variational autoencoders for robust speech recognition.

[20] Yu, D., Li, J., Xu, L., & Deng, L. (2017). Joint learning of speech enhancement and automatic speech recognition with deep neural networks.