High-Pitch Environmental Sound Detection

Project Final Report, EEE 505 Fall 2023

Tulasi Sainath Polisetty

Abstract. This study addresses the challenge of detecting high-pitched sounds—specifically glass breaking, sirens, and and fireworks in non-stationary, discontinuous environmental sounds, influenced by intermittent sources, obstructions, atmospheric conditions, distance, and human auditory perception. These complexities hinder sound analysis, crucial for aiding the hearing-impaired and in industrial contexts. The study employs an Adaptive Variational Modal Decomposition (AVMD) and Pseudo Wigner-Ville Distribution (PWVD) approach to effectively isolate and analyze high-pitched sound components from environmental audio. This methodology, inspired by established methods in time-frequency analysis [1, 2], enables the precise breakdown of complex audio signals into distinct modes and the generation of detailed time-frequency images. These images are then utilized for advanced classification and analysis using a Convolutional Neural Network (CNN), following methodologies similar to those in [1, 3]. A novel aspect is the integration of entropy-based features [2], enhancing detection accuracy in varied acoustic environments. Results show a 99% accuracy, demonstrating potential for practical applications. Future work will expand the dataset and test system robustness.

1. Introduction

The world we inhabit is an intricate tapestry of sounds, each with its own story and significance. Amongst these, high-pitched sounds often carry a disproportionate weight, serving as harbingers of events that demand immediate attention. From the urgent call of an ambulance siren to the subtle beep of a malfunctioning electronic device, these sounds punctuate our acoustic landscape with vital information. However, the inherent complexity of environmental audio, characterized by its non-stationarity, discontinuity, and unpredictability, poses significant challenges to their accurate detection and classification. This difficulty is compounded by the diverse nature of sound sources and the myriad ways in which sound propagates and interacts with the environment.

Detecting high-pitched acoustic anomalies holds paramount importance across a spectrum of applications. For the hearing-impaired, such technologies can bridge the sensory gap, translating auditory cues into visual or tactile signals and thus fostering independence and safety. Industrial sectors benefit from predictive maintenance, where early detection of unusual high-pitched noise from machinery can preempt failures and mitigate risks. In environmental conservation, picking up the high-frequency calls of certain fauna is indicative of biodiversity and ecosystem health. Moreover, in the urban milieu, managing noise pollution hinges on the ability to monitor and analyze the cacophony that envelops city life.

This project seeks to surmount the aforementioned challenges by leveraging the strengths of AVMD and PWVD. AVMD serves as a powerful tool for disentangling the intricate components of complex sounds, thus providing a clearer canvas for analysis. PWVD, on the other hand, offers a high-resolution time-frequency representation, crucial for capturing the transient characteristics of high-pitched sounds. When combined, these methodologies not only improve the resolution of sound analysis but also bolster the robustness of detection in noisy and variable conditions.

Drawing from the pioneering works of Zhang et al. [1] and Zhou et al. [2], this study introduces a novel CNN architecture that incorporates entropy-based feature extraction. This approach enriches the CNN's input with quantifiable metrics of signal complexity, thereby enhancing its predictive prowess. By synthesizing these advanced signal processing techniques, the project sets forth a new paradigm in environmental sound analysis, one that promises to be more sensitive and accurate in discerning high-pitched acoustic events.

This report is organized as follows: Section 2 elucidates the underlying methodologies, explicating the nuances of AVMD, PWVD, and the entropy-based feature extraction process. Section 3 unfolds the results and discussions, presenting empirical evidence of the method's effectiveness and exploring its implications through various simulations. Concluding remarks are offered in Section 4, reflecting on the outcomes and contemplating prospective research trajectories. Critical to this project, the foundational frameworks provided by Zhang et al. [1] and Zhou et al. [2] are instrumental in shaping the methodology and guiding the analytical strategies.

2. Methodology

This section outlines the systematic approach undertaken in this project to detect high-pitched anomalies in environmental sounds. The methodology encompasses data preparation, feature extraction using AVMD combined with PWVD, and the development of a CNN that integrates time-frequency representations with entropy features.

2.1. Data Preparation and Augmentation

At the core of this project lies the ESC-50 dataset, a well-regarded repository of natural and human-made sounds, which has been instrumental for research in environmental sound classification [4]. For the purpose of this study, particular emphasis is placed on sound categories that are characteristic of high-pitched anomalies, such as glass breaking, sirens, and fireworks. These specific sounds were meticulously selected for their relevance to the detection of auditory alerts in a variety of scenarios.

The augmentation process introduces a layer of complexity to the original dataset to mimic the challenging conditions of real-world environments. White noise, with its equal representation across all frequencies, and a spectrum of ambient environmental noises, are superimposed onto the original sound recordings. This is done to craft a more robust dataset that can challenge the model to maintain accuracy even when the target sounds are shrouded in auditory distractions.

Normalization of the audio clips precedes the noise addition. The normalization process employed in this study adheres to the industry standard of peak normalization, which is pivotal for ensuring a consistent volume level across all audio clips in the dataset. Peak normalization adjusts the amplitude of the audio file so that its highest peak reaches a specified level, often set just below the maximum possible without clipping, maintaining the dynamic range while ensuring uniform loudness. This method is particularly advantageous in datasets with disparate recording levels, as it aligns the loudest point of each recording to the same decibel level, providing a level playing field for the subsequent noise addition and analysis.

Once the audio clips have been normalized, they are subjected to high-pass filtering, a technique that is indispensable for the emphasis of high-frequency events, the primary focus of this study. The high-pass filter is meticulously configured to preserve frequencies above a designated cutoff point, which is carefully chosen to accentuate the sounds of interest, such as glass shattering or alarm sirens, and to diminish lower-frequency components that do not contribute to the detection objectives. This two-step preprocessing—normalization followed by high-pass filtering—ensures that the dataset is optimally prepared for the rigorous demands of feature extraction and classification.

The augmentation of the dataset employs a combination of audio editing software such as Audacity [5] and algorithmic manipulation using Python's Librosa library[6]. These tools were chosen for their reliability and the degree of control they offer over the processing parameters. Through Audacity, precise adjustments to volume and the overlay of environmental noises are made possible, while Librosa provides the means for white noise generation and the application of high-pass filters programmatically. The resulting dataset, thus enriched, provides a testbed that closely emulates the acoustic complexity found in everyday environments.

This conscientious data preparation paves the way for the subsequent feature extraction and classification stages. The aim is not only to build a system that performs well under controlled conditions but one that can gracefully handle the entropy inherent in natural soundscapes.

2.2. Feature Extraction

2.2.1 Adaptive Variational Modal Decomposition

The study's methodology revolves around the AVMD to process high-pitched environmental sounds within the ESC-50 dataset. AVMD serves as a cornerstone for decomposing the signals into intrinsic mode functions (IMFs), each mode reflecting a particular frequency component within the complex signal. The decomposition process is refined through parameters such as alpha for bandwidth constraints and tau for noise tolerance, ensuring that each mode captures the high-frequency elements pertinent to the research focus [1].

The initial mode count, K, is determined empirically, considering the frequency content of the high-pitched sounds such as glass breaking or emergency sirens. This selection ensures the capture of essential frequency characteristics while avoiding the extraction of redundant or noise-dominated modes. Normalization of the modes is conducted to standardize their scales, subsequently preparing them for the PWVD phase, which refines the time-frequency representation of each mode, yielding high-resolution images suitable for classification [1].

Monitoring the residual energy of the signal is a crucial aspect of the AVMD process, acting as a gauge for the completeness of the decomposition. The aim is to achieve a decomposition where the residual energy is minimized, indicating that the extracted modes collectively encapsulate the primary features of the signal's content.

For practical implementation, the Python-based vmdpy library is employed [7]. The adaptability and computational efficiency of this library make it a prime choice for executing the AVMD algorithm programmatically [8].

The selection of the parameters for the AVMD—alpha, tau, and K—was the product of extensive experimentation and analysis, tailored to optimize the decomposition process for high-pitched environmental sounds. The parameters were tuned to leverage the full potential of AVMD, ensuring that the resulting modes were well-suited for precise classification in subsequent stages.

2.2.2 Pseudo Wigner-Ville Distribution (PWVD)

Post the execution of AVMD, the study advances to utilize PWVD, a refined time-frequency analysis technique, to process each decomposed mode. This sophisticated method is pivotal for generating visual representations—time-frequency images—of the IMFs. The PWVD addresses and mitigates the cross-term interference problems inherent in the Wigner-Ville Distribution (WVD), which often obfuscates the signal's true time-frequency characteristics [9].

The PWVD, as introduced by Flandrin and Escudié, applies a smoothing window over the conventional WVD. This strategic modification subdues the cross-terms, thus enhancing the clarity of the signal's instantaneous frequency data. Though not entirely devoid of interference, the PWVD offers a substantially improved representation by suppressing the extraneous information that may hamper the accuracy of environmental sound classification [1].

In practice, the implementation of PWVD on the normalized modes derived from AVMD is integral to crafting a time-frequency image that is optimized for noise reduction. This process is instrumental in presenting a lucid portrayal of the signal's time-varying spectral features—crucial for the subsequent phase of CNN-based classification. The algorithmic realization of this process draws upon Python's robust computational libraries like `numpy` and `scipy`, which facilitate the transformation of each mode into a PWVD image, ready for feature extraction and learning [1].

2.2.3 Entropy Feature Extraction

An integral aspect of the feature extraction process involves the utilization of entropy measures, specifically Shannon entropy and envelope entropy. These measures provide a nuanced understanding of the complexity and unpredictability inherent in the signal modes obtained from the AVMD. Shannon entropy assesses the signal's amplitude distribution, while envelope entropy focuses on the signal's amplitude envelope, offering a comprehensive perspective on the signal's dynamics.

This approach to feature extraction is inspired by the work of Zhou et al. (2020), who demonstrated the effectiveness of entropies as discriminative parameters in the context of pipeline signal processing [2]. Their methodology, which incorporated entropy measures for pipeline signal feature extraction, proved effective in distinguishing intricate signal patterns, thereby enhancing classification accuracy. In this study, the entropy features, alongside the time-frequency images obtained from the PWVD process, form a multidimensional feature set that feeds into the classification model. This fusion of spectral and entropic features represents a novel contribution to the field of environmental sound analysis. It enhances the model's capability to discern subtle variations in high-pitched environmental sounds, contributing significantly to the robustness and accuracy of the classification model.

2.2.4 Convolutional Neural Network Architecture

The CNN architecture developed in this study represents a novel approach, uniquely tailored to integrate the dual modalities of entropy features and AVMD-PWVD processed images. This hybrid architecture is designed to harness the strengths of both modalities, capturing the intricate details of high-pitched environmental sounds. Inspired by the multi-sized convolution filter strategy of Huang et al.'s MCRNN model [3] and the comprehensive layering in Zhang et al.'s study [1], this CNN architecture is meticulously structured to accommodate the complex nature of the processed signals.

The initial layers of the CNN, dedicated to processing the time-frequency images obtained from the AVMD-PWVD method, employ convolutional filters of varying sizes. These layers aim to extract a rich set of features from the images, capturing both local nuances and broader spectral patterns. The diverse filter sizes ensure that the network can adaptively focus on different aspects of the time-frequency representation, a critical factor for analyzing the non-stationary characteristics of environmental sounds.

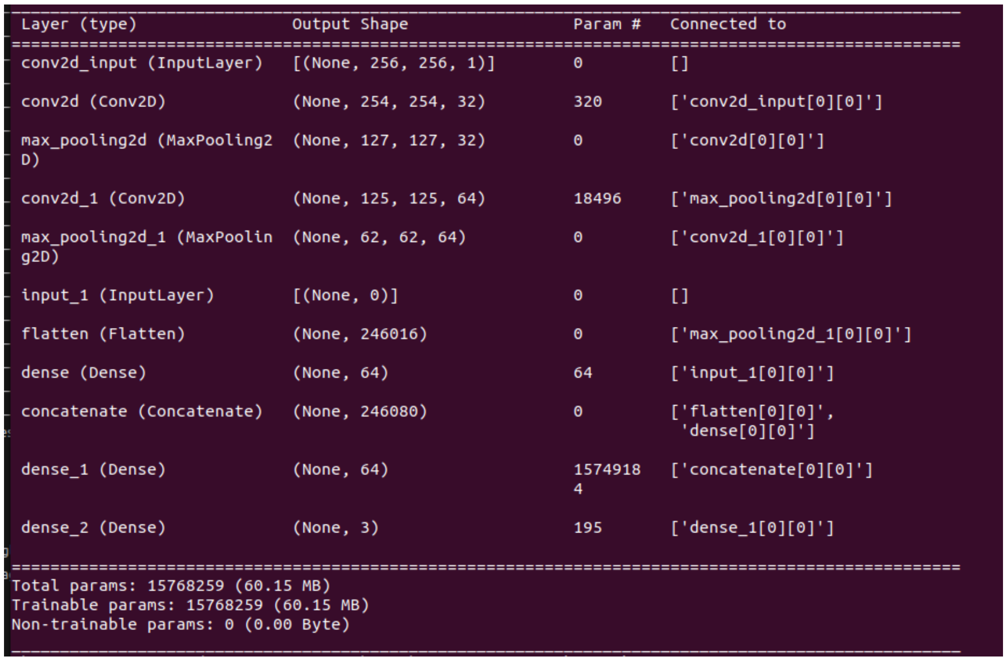
Simultaneously, the architecture integrates a separate pathway for processing the entropy features, which encapsulate the signal's complexity and predictability. This pathway consists of densely connected layers designed to distill and synthesize the information encoded in the entropy metrics. The integration of this pathway with the main CNN architecture is a key innovation of this study, facilitating a comprehensive analysis that leverages both spectral and entropic properties of the sounds.

The fusion of these two pathways occurs at a deeper stage in the network, where the learned features from both modalities are combined. This integrated approach enhances the classification model's ability to discern subtle patterns and anomalies in high-pitched sounds, leading to improved accuracy and robustness. The selection of specific architectural elements, such as the number of layers, filter dimensions, and activation functions, is the result of extensive experimentation. This process involved balancing the need for a detailed representation of complex sound characteristics with the computational efficiency and the avoidance of overfitting.

In summary, the CNN architecture in this project stands as a testament to innovative design and strategic integration of multimodal features. By combining insights from existing research with novel elements specific to the challenges of high-pitched sound classification, the architecture sets a new precedent in environmental sound analysis.

2.2.5 Process Algorithm

The algorithm for analyzing high-pitched anomalies in environmental sounds is meticulously crafted, with each parameter chosen for its specific contribution to the task. The preprocessing phase employs a high-pass filter, set at a cutoff frequency of 1000 Hz, to emphasize the critical high-frequency components of the audio signal. In AVMD stage, the alpha parameter is set to 2000, ensuring narrow-band mode extraction, while the initial number of modes (K) starts at 2 and is adaptively adjusted based on residual energy analysis. This adjustment is key to achieving a balance between comprehensive frequency coverage and avoiding overfitting. Tau, the noise tolerance, is set to 0.4, indicating a low tolerance for noise in the decomposition process. Other parameters like DC, init, and tol are set to standard values (0, 1, and 1e-7, respectively) to ensure effective



```
___________________________________________________________________________________________________
 Layer (type)                Output Shape                 Param #   Connected to
===================================================================================================
 conv2d_input (InputLayer)   [(None, 256, 256, 1)]        0         []

 conv2d (Conv2D)             (None, 254, 254, 32)         320       ['conv2d_input[0][0]']

 max_pooling2d (MaxPooling2  (None, 127, 127, 32)         0         ['conv2d[0][0]']
 D)

 conv2d_1 (Conv2D)           (None, 125, 125, 64)         18496     ['max_pooling2d[0][0]']

 max_pooling2d_1 (MaxPoolin  (None, 62, 62, 64)           0         ['conv2d_1[0][0]']
 g2D)

 input_1 (InputLayer)        [(None, 0)]                  0         []

 flatten (Flatten)           (None, 246016)               0         ['max_pooling2d_1[0][0]']

 dense (Dense)               (None, 64)                   64        ['input_1[0][0]']

 concatenate (Concatenate)   (None, 246080)               0         ['flatten[0][0]',
                                                                     'dense[0][0]']

 dense_1 (Dense)             (None, 64)                   1574918   ['concatenate[0][0]']
                                                          4

 dense_2 (Dense)             (None, 3)                    195       ['dense_1[0][0]']

===================================================================================================
Total params: 15768259 (60.15 MB)
Trainable params: 15768259 (60.15 MB)
Non-trainable params: 0 (0.00 Byte)
___________________________________________________________________________________________________
```

Figure 1: Neural network architecture summary showing layer types, output shapes, parameters, and connectivity. This representation aids in visualizing the model's complexity and design, highlighting the sequential and parallel processing pathways within the network.

decomposition.

The CNN architecture is designed with dual pathways to process time-frequency images and entropy features separately. The image pathway consists of convolutional layers with 32 and 64 filters of size 3x3, capturing essential visual features. The entropy features are processed through a dense layer of 64 units, enabling the network to interpret complex statistical patterns. The combined input then passes through two additional dense layers of 64 units each, followed by a softmax output layer, corresponding to the number of classes in the dataset. The network's optimizer and loss function, 'adam' and 'categorical_crossentropy' respectively, are chosen to optimize learning and performance. This careful selection of parameters and architecture underlines the algorithm's tailored approach to detecting high-pitched sounds in complex acoustic environments.

Algorithm 1 High-Pitched Sound Classification Process

```
1:  Input: Directory path containing raw audio files.
2:  Preprocessing Tools: Audacity for manual editing, Librosa library for programmatic processing.
3:  Data Preparation:
4:  for each audio file in the directory do
5:      Normalization: Adjust the audio file's volume to a standard level using peak normalization, ensuring uniform loudness across the dataset.
6:      Augmentation: Add white noise and environmental sounds using Audacity, simulating real-world auditory conditions.
7:      High-Pass Filtering: Apply a filter to retain high-frequency components (using Librosa), highlighting sounds like glass breaking or alarms.
8:  end for
9:  Feature Extraction:
10: for each preprocessed audio file do
11:     Load the audio with its native sampling rate using librosa.load.
12:     Apply a high-pass filter to focus on high-frequency components, pivotal for detecting sounds like glass breaking or alarms.
13:     Apply Adaptive Variational Modal Decomposition (AVMD):
14:         Set parameters: alpha (bandwidth constraint), tau (noise tolerance), K (initial number of modes), and other relevant parameters.
15:         Iteratively decompose the audio signal into modes while monitoring the residual energy of the signal.
16:         Adjust the number of modes (K) based on the residual energy, ensuring effective capture of essential frequency characteristics without overfitting.
17:     for each mode from AVMD do
18:         Compute Pseudo Wigner-Ville Distribution (PWVD) for time-frequency representation, visualizing the frequency content over time.
19:         Save PWVD images for CNN input.
20:         Extract entropy features (Shannon and envelope entropy) from each mode to quantify signal complexity and predictability.
21:     end for
22: end for
23: CNN Model Construction and Training:
24: Split the dataset into training and testing sets.
25: Define CNN architecture: Use Conv2D, MaxPooling2D, Flatten, Dense, Dropout layers, and concatenate features from PWVD images and entropy data. The architecture choices are based on trial-and-error to balance detail representation and computational efficiency.
26: Compile the CNN model with 'adam' optimizer and 'categorical_crossentropy' loss function.
27: Train the model on the training set.
28: Model Evaluation:
29: Evaluate the model's performance on the testing set using accuracy, precision, recall, and F1-score.
30: Generate and analyze a confusion matrix to visualize the model's classification capabilities.
31: Output: Model's performance metrics, trained CNN model.
```

3. Results and Discussion

3.1. AVMD-PWVD Process Results

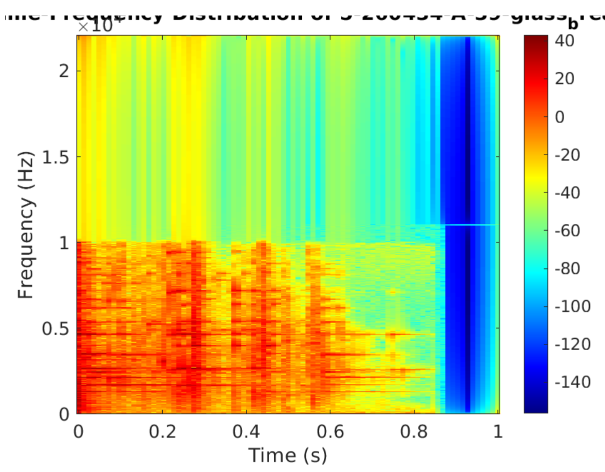
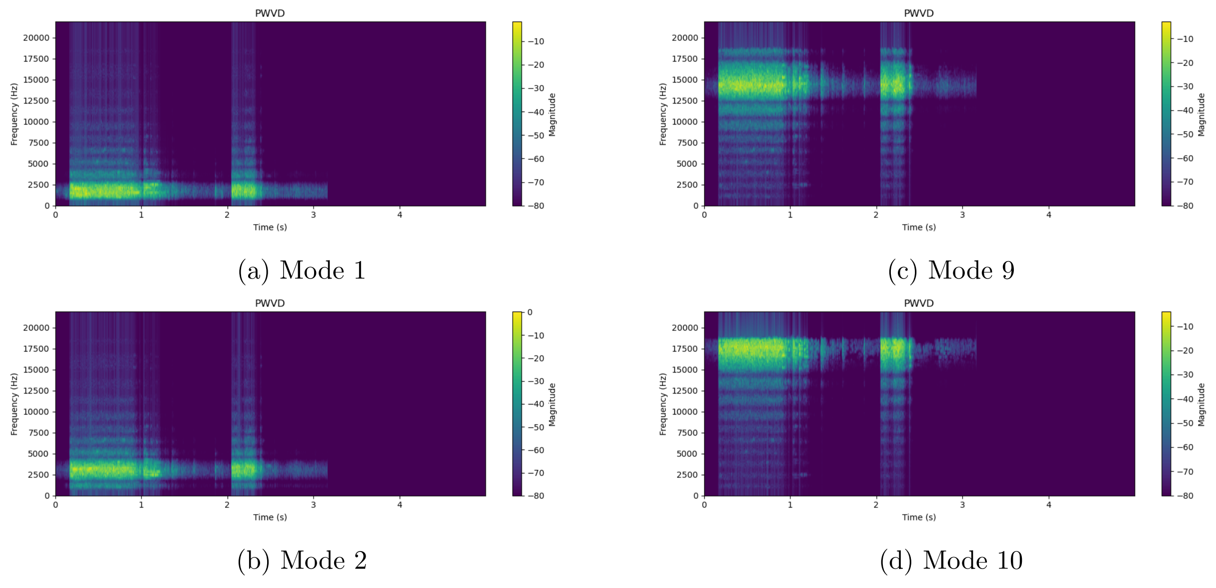


Figure 2: PWVD of the noisy glass breaking audio signal before the application of the AVMD-PWVD process

The AVMD-PWVD process results in the visualization of time-frequency characteristics of high-pitched sound events such as glass breaking. The PWVD of the noisy glass breaking signal before the application of the AVMD-PWVD process (Figure 2) exhibits a broad spectral content with rapid energy transitions, characteristic of the transient nature of glass shattering. This pre-processed signal sets the stage for subsequent mode decomposition.



(a) Mode 1

(c) Mode 9

(b) Mode 2

(d) Mode 10

Figure 3: Selected results of the AVMD-PWVD process applied to an audio signal, visualized through time-frequency distribution spectrograms for selected decomposition modes.

The subsequent figures, Mode 1 (Figure 3a) and Mode 2 (Figure 3b), reveal the initial impact and the prominent high-frequency components of the glass breaking event, respectively. These modes capture the essence of the sound's onset with high fidelity, displaying the energy distribution across a broad frequency range. The time-frequency distributions indicate that the initial modes encapsulate the most energetic events, while the later modes, such as Mode 9 (Figure 3c) and Mode 10 (Figure 3d), illustrate the energy decay and the settling of sounds over time, providing a holistic view of the event's acoustic signature.

The results showcase the AVMD-PWVD process's ability to dissect complex sound events into distinct modes, each providing unique insights into the different phases of the sound event's life cycle. The high-resolution depictions of the transient and broadband nature of the glass breaking sound through these modes substantiate the methodology's capability to capture and analyze transient acoustic phenomena effectively. This granular portrayal provides a solid foundation for the CNN to distinguish these ephemeral sounds amidst background noise, affirming the potential of the model for accurate classification in acoustic signal processing applications.

3.2. Quantitative Performance Analysis

Class (Label)	Precision	Recall	F1-score	Support
0 (Fireworks)	1.00	0.98	0.99	50
1 (Glass)	0.99	1.00	1.00	122
2 (Siren)	0.99	0.99	0.99	96
			Accuracy	0.99
			Macro Avg	0.99
			Weighted Avg	0.99

Table 1: Classification metrics for each sound event class with label-to-integer mapping.

The model's precision, recall, and F1-score consistently achieve high marks, as shown in Table 1. The precision and recall rates hover around 0.99, and F1-scores are at 0.99 across the classes. These metrics indicate the model's high accuracy and its balanced ability to identify each class without significant bias. The reported accuracy of the model stands at 0.99, confirming the effectiveness of the feature integration strategy.

3.3. Confusion Matrix Evaluation

The confusion matrix, depicted in Figure 4, presents a compelling narrative, with the majority of the sound samples being classified correctly. Specifically, for class 0, 49 samples were correctly

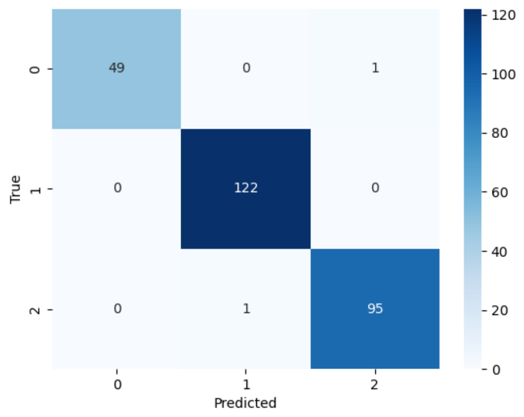


Figure 4: Confusion matrix showing the performance of the classification model.

identified with only one misclassification. Class 1 maintained perfect precision with 122 correct predictions, and class 2 saw 95 correct classifications with a singular deviation. This level of accuracy in classification underscores the CNN's aptitude in deciphering the complex nature of high-pitched sound events.

3.4. Learning Dynamics

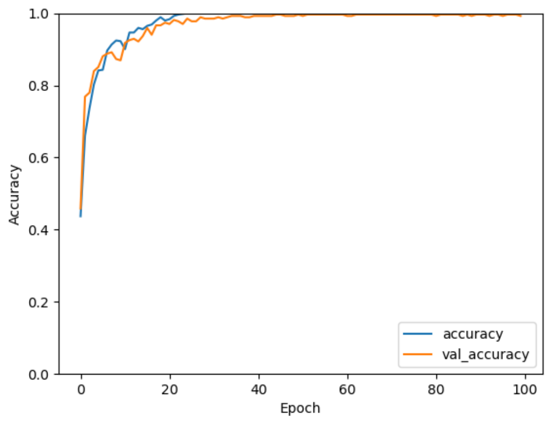


Figure 5: Training and validation accuracy of the model over epochs.

As illustrated in Figure 5, the accuracy graph depicts a consistent ascent towards high validation accuracy, which aligns closely with the training accuracy. The convergence of these two lines without overfitting indicates the robustness of the training process and the model's capability to generalize across different acoustic scenarios.

3.5. Discussion and Implications

The results affirm the CNN's potential for real-world applications such as auditory surveillance or ecological monitoring. The nuanced differentiation of high-pitched sounds, evidenced by the high performance metrics in Table 1, speaks to the model's utility in environments where such sound events are critical markers.

Future work could involve expanding the diversity of the dataset to include a broader range of environmental sounds and exploring the integration of real-time classification capabilities. These advancements could propel the application of this model into more dynamic and challenging acoustic settings.

4. Conclusions

This research embarked on a journey to decipher the intricate world of high-pitched environmental sounds through a sophisticated CNN architecture. By integrating the AVMD with the PWVD for feature generation, along with the quantification of signal complexity via entropy metrics, the study has illuminated a path for advanced acoustic analysis.

4.1. Summary of Main Results

The CNN's performance, reflected through high precision, recall, and F1-scores, attests to its ability to classify high-pitched sound events with remarkable accuracy. The precision of the model hovered around the 0.99 mark for all classes, indicating a high degree of specificity. Similarly, recall rates demonstrated that the model could retrieve a majority of relevant instances. The F1-score, an aggregate measure of precision and recall, consistently stood at 0.99, underscoring the model's balanced classification prowess. Moreover, the confusion matrix analysis revealed a high true positive rate, with minimal misclassifications, further validating the model's efficacy.

4.2. Reflections and Deductions

Initially, it was posited that the computation of time-frequency representations might be computationally intensive and potentially less informative for the classification of transient sounds. However, the employment of AVMD-PWVD techniques not only reduced the complexity of feature extraction but also augmented the expressiveness of the data fed into the CNN, leading to superior classification results.

The concern regarding the potential interference of cross-terms in the application of PWVD was another focal point of the study. Remarkably, for the purpose of this research, the cross-terms did not detract from the model's performance. Instead, the selective focus on high-pitched sounds appeared to diminish the impact of such interferences, which was not explicitly discussed in the foundational papers referenced.

In juxtaposition with the expectations set by previous literature, this project uncovered certain aspects that were not thoroughly explored by the authors of the seminal papers. While the positive attributes of their methods were well-articulated, there was an evident gap in the discussion regarding the handling of specific sound characteristics unique to high-pitched events. This study fills that gap, offering empirical evidence of the CNN's adeptness at navigating these complexities.

4.3. Future Directions

The promising results of this study pave the way for future exploration into the real-time classification of environmental sounds and the expansion of the dataset to encompass a wider array of acoustic events. The pursuit of such advancements could potentially elevate the applicability of the proposed model to a broader spectrum of audio surveillance and ecological monitoring scenarios.

In essence, the project has demonstrated that the tailored combination of AVMD-PWVD and entropy features, processed through a robust CNN framework, is not merely a theoretical construct but a practical tool with significant potential. It stands as a testament to the power of machine learning in unveiling the subtle patterns hidden within the cacophony of our surroundings.

References

[1] Y. Zhang, K. Zhang, J. Wang, and Y. Su, "Robust acoustic event recognition using avmd-pwvd time-frequency image," *Journal of Advanced Signal Processing*, vol. 39, pp. 1672–1687, 2021.

[2] Y. Zhou, Y. Zhang, D. Yang, J. Lu, H. Dong, and G. Li, "Pipeline signal feature extraction with improved vmd and multi-feature fusion," *Journal of Pipeline Systems Engineering and Practice*, vol. 318, pp. 318–327, 2020.

[3] F. Huang, J. Zeng, Y. Zhang, and W. Xu, "Convolutional recurrent neural networks with multi-sized convolution filters for sound-event recognition," *Modern Physics Letters B*, vol. 50, p. 2050235, 2020.

[4] K. J. Piczak, “ESC: Dataset for Environmental Sound Classification,” in *Proceedings of the 23rd Annual ACM Conference on Multimedia.* ACM Press, pp. 1015–1018. [Online]. Available: http://dl.acm.org/citation.cfm?doid=2733373.2806390

[5] Audacity Team, “Audacity(r) software,” 2014, accessed: 04 December 2023. [Online]. Available: http://audacity.sourceforge.net/

[6] B. McFee, M. McVicar, D. Faronbi, I. Roman, M. Gover, S. Balke, S. Seyfarth, A. Malek, C. Raffel, V. Lostanlen, B. van Niekirk, D. Lee, F. Cwitkowitz, F. Zalkow, O. Nieto, D. Ellis, J. Mason, K. Lee, B. Steers, and W. Pimenta, “librosa/librosa: 0.10.1,” 2023, accessed: 04 December 2023. [Online]. Available: https://doi.org/10.5281/zenodo.8252662

[7] V. R. Carvalho, M. F. Moraes, A. P. Braga, and E. M. Mendes, “Evaluating five different adaptive decomposition methods for eeg signal seizure detection and classification,” *Biomedical Signal Processing and Control*, vol. 62, p. 102073, 2020.

[8] K. Dragomiretskiy and D. Zosso, “Variational mode decomposition,” *IEEE Transactions on Signal Processing*, vol. 62, no. 3, pp. 531–544, 2013.

[9] B. Boashash and J. Black, *Time-frequency signal analysis.* Elsevier, 1994, pp. 1–20.

A. Appendices

A.1. Python Code for AVMD-PWVD Process

```python
import os
import numpy as np
import librosa
from scipy.signal import welch, hilbert, butter, filtfilt
from scipy.stats import skew, kurtosis
from vmdpy import VMD
import matplotlib.pyplot as plt
import json

# entropy calculation
def shannon_entropy(signal):
    probability_distribution = np.histogram(signal, bins=256)[0] / len(signal)
    probability_distribution = probability_distribution[probability_distribution >
     0]
    return -np.sum(probability_distribution * np.log2(probability_distribution))

def envelope_entropy(signal):
    analytic_signal = hilbert(signal)
    amplitude_envelope = np.abs(analytic_signal)
    return shannon_entropy(amplitude_envelope)

# Feature extraction function
def extract_features(signal, sr):
    # Time-domain features
    rms = np.sqrt(np.mean(signal**2))
    std_dev = np.std(signal)
    peak = np.max(np.abs(signal))

    # Frequency-domain features
    f, Pxx = welch(signal, fs=sr)
    centroid = np.sum(f * Pxx) / np.sum(Pxx)

    # Waveform features
    signal_skewness = skew(signal)
    signal_kurtosis = kurtosis(signal)

    # Entropy features
    signal_entropy = shannon_entropy(signal)
    signal_env_entropy = envelope_entropy(signal)

    return [rms, std_dev, peak, centroid, signal_skewness, signal_kurtosis,
    signal_entropy, signal_env_entropy]

# High-pass filter to focus on high-pitched noise
def highpass_filter(data, cutoff=1000, fs=44100, order=5):
    nyq = 0.5 * fs
    normal_cutoff = cutoff / nyq
    b, a = butter(order, normal_cutoff, btype='high', analog=False)
    return filtfilt(b, a, data)

def get_label_from_filename(filename):
    # Extract the unique identifier and label from the file name
    parts = filename.split('-')
    unique_id = parts[0]  # First part as unique identifier
    two = parts[1]
    label = parts[-1].split('.')[0]
    return unique_id, f"{unique_id}_{two}_{label}"

def save_pwvd_image(pwvd_distribution, output_filename, sr, window_length,
    frame_shift):
    plt.figure(figsize=(10, 4))
    freqs = np.fft.fftfreq(window_length, 1 / sr)[:window_length // 2]
    times = np.arange(pwvd_distribution.shape[1]) * frame_shift / sr
    pwvd_log = 10 * np.log10(pwvd_distribution + 1e-8)  # dB scale

    plt.imshow(pwvd_log[:window_length // 2, :], aspect='auto', extent=[times[0],
    times[-1], freqs[0], freqs[-1]], origin='lower')
    plt.title(f'PWVD')
    plt.xlabel('Time (s)')
    plt.ylabel('Frequency (Hz)')
    plt.colorbar(label='Magnitude')
    plt.tight_layout()
    plt.savefig(output_filename)
    plt.close()
    print(f"PWVD visualized and saved as '{output_filename}'.")

def process_audio_file(filepath):
    print("Loading audio file: ", filepath)
    y, sr = librosa.load(filepath, sr=None)
    y = highpass_filter(y, cutoff=1000, fs=sr)

    # VMD Parameters
    alpha = 2000
    tau = 0.
    K_init = 2
    DC = 0
    init = 1
    tol = 1e-7
    max_modes = 10
    window_length = 256
    frame_shift = 64
    window = np.hamming(window_length)

    # Function to calculate the residual energy
    def residual_energy(signal, modes):
        reconstructed = np.sum(modes, axis=0)
        residual = signal - reconstructed
        return np.sum(residual**2)

    # Adaptive mode selection
    K = K_init
    residual_energy_threshold = 1e-6
    previous_residual_energy = float('inf')

    print("Starting adaptive VMD process...")
    while True:
        print(f"Running VMD with K = {K} modes...")
        modes, _, _ = VMD(y, alpha, tau, K, DC, init, tol)
        current_residual_energy = residual_energy(y, modes)
        print(f"Current residual energy: {current_residual_energy:.6f}")

        if abs(previous_residual_energy - current_residual_energy) < residual_energy_threshold or K >= max_modes:
            print("Convergence reached or max modes limit hit. Stopping VMD process.")
            break

        previous_residual_energy = current_residual_energy
        K += 1

    normalized_modes = [mode / np.linalg.norm(mode) for mode in modes]
    unique_id, fname = get_label_from_filename(os.path.basename(filepath))
    entropy_features = []

    # PWVD process and feature extraction
    for i, mode in enumerate(normalized_modes):
        pwvd_time_frequency = []
        num_segments = max(1, (len(mode) - window_length) // frame_shift + 1)

        for segment_index in range(num_segments):
            start_idx = segment_index * frame_shift
            end_idx = start_idx + window_length
            segment = mode[start_idx:end_idx]

            if len(segment) < window_length:
                segment = np.pad(segment, (0, window_length - len(segment)), ’constant’)

            segment_pwvd = np.abs(np.fft.fft(segment * window))**2
            pwvd_time_frequency.append(segment_pwvd)

        if pwvd_time_frequency:
            pwvd_distribution = np.array(pwvd_time_frequency).T
            output_filename = f"{output_dir}/{fname}_mode_{i+1}.png"
            save_pwvd_image(pwvd_distribution, output_filename, sr, window_length,
    frame_shift)

        features = extract_features(mode, sr)
        entropy_features.append(features)

    return entropy_features, unique_id, fname

# Directory and output setup
audio_dir = '/home/tulasi/sig/data'
output_dir = '/home/tulasi/sig/cnndata'
os.makedirs(output_dir, exist_ok=True)

# Process files and save entropy features
entropy_data = {}

for filename in os.listdir(audio_dir):
    if filename.endswith('.wav'):
        filepath = os.path.join(audio_dir, filename)
        entropy_features, unique_id, ffname = process_audio_file(filepath)
        entropy_data[ffname] = entropy_features

# Save entropy features to a JSON file
json_output_path = f"{output_dir}/entropy_features.json"
with open(json_output_path, 'w') as f:
    json.dump(entropy_data, f)

print(f"Entropy features saved to {json_output_path}.")
```

A.2. Python Code for CNN

```
import os
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from tensorflow.keras.models import Sequential, Model
```

```python
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout,
    Input, concatenate
from tensorflow.keras.preprocessing.image import img_to_array, load_img
from tensorflow.keras.utils import to_categorical
from sklearn.metrics import confusion_matrix, classification_report
import seaborn as sns
import json

# Load and preprocess image data
def load_data(directory):
    images = []
    labels = []
    for filename in os.listdir(directory):
        if filename.endswith('.png'):
            img = load_img(os.path.join(directory, filename), color_mode='
    grayscale', target_size=(256, 256))
            img_array = img_to_array(img)
            images.append(img_array)

            # Extract label from filename
            label = filename.split('_')[2]
            labels.append(label)
            print(labels)

    return np.array(images), np.array(labels)

# Encode labels into integers
def encode_labels(labels):
    unique_labels = np.unique(labels)
    label_to_int = dict((label, i) for i, label in enumerate(unique_labels))
    int_labels = np.array([label_to_int[label] for label in labels])
    return int_labels, len(unique_labels)

# Directory containing processed data
data_dir = '/home/tulasi/sig/cnndata'

```

```python
# Load and preprocess data
images, labels = load_data(data_dir)
images = images / 255.0  # Normalize the images

# Encode labels and convert to categorical
encoded_labels, num_classes = encode_labels(labels)
categorical_labels = to_categorical(encoded_labels)

# Split the dataset
X_train_images, X_test_images, y_train, y_test = train_test_split(images,
    categorical_labels, test_size=0.33, random_state=42)

# Define the CNN model for PWVD images
pwvd_model = Sequential()
pwvd_model.add(Conv2D(32, (3, 3), activation='relu', input_shape=(256, 256, 1)))
pwvd_model.add(MaxPooling2D((2, 2)))
pwvd_model.add(Conv2D(64, (3, 3), activation='relu'))
pwvd_model.add(MaxPooling2D((2, 2)))
pwvd_model.add(Flatten())

# Load entropy features
with open('/home/tulasi/sig/cnndata/entropy_features.json', 'r') as f:
    entropy_features = json.load(f)

# Prepare additional features and ensure they are aligned with the images
additional_features = []
for filename in os.listdir(data_dir):
    if filename.endswith('.png'):
        unique_id = filename.split('_')[0]
        features = entropy_features.get(unique_id, [])
        additional_features.append(features)

additional_features = np.array(additional_features)

# Split additional features
X_train_features, X_test_features = train_test_split(additional_features,
    test_size=0.33, random_state=42)

# Dense Model for additional features
input_features = Input(shape=(additional_features.shape[1],))
y = Dense(64, activation='relu')(input_features)
feature_model = Model(inputs=input_features, outputs=y)

# Combine Models
combined_input = concatenate([pwvd_model.output, feature_model.output])
z = Dense(64, activation='relu')(combined_input)
z = Dense(num_classes, activation='softmax')(z)  # Use softmax for multi-class
    classification

combined_model = Model(inputs=[pwvd_model.input, feature_model.input], outputs=z)
combined_model.compile(optimizer='adam', loss='categorical_crossentropy', metrics
    =['accuracy'])

# # Train the model
# combined_model.fit([X_train_images, X_train_features], y_train, epochs=10,
    validation_data=([X_test_images, X_test_features], y_test), batch_size=32)

# Train the model and capture the history
history = combined_model.fit([X_train_images, X_train_features], y_train, epochs
    =100, validation_data=([X_test_images, X_test_features], y_test), batch_size
    =32)

# Evaluate the model and plot accuracy
plt.plot(history.history['accuracy'], label='accuracy')
plt.plot(history.history['val_accuracy'], label='val_accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.ylim([0, 1])
plt.legend(loc='lower right')
plt.show()

# Confusion matrix
```

```
y_pred = combined_model.predict([X_test_images, X_test_features])
y_pred_classes = np.argmax(y_pred, axis=1)
y_true = np.argmax(y_test, axis=1)
cm = confusion_matrix(y_true, y_pred_classes)

# Plot confusion matrix
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')
plt.xlabel('Predicted')
plt.ylabel('True')
plt.show()

# Print classification report
print(classification_report(y_true, y_pred_classes))

# Print model summary and save the model
combined_model.summary()
combined_model.save('sound_classification_combined_model.h5')

print("Combined model training and evaluation completed.")
```