Paper 1 and Paper 2 are the papers that i had for analyzing and creating an idea of what has been done till now , the experiments and an overview of what can be done. It is summarized in this file.

For reading about what I have done go to page → 26

Paper 1

When we speak, it's not just about making sounds—it's a complex process involving several key components:

1. **Excitation Source**: This is like the engine of speech. It could be a sound we make with our vocal cords (like when we hum), or it could be silence (like during a pause).
2. **Articulation**: This is how we shape those sounds using our mouth, tongue, and other parts of the vocal tract to create specific sounds, like different vowels and consonants.
3. **Fluency**: This refers to how smoothly and quickly we speak.

Speech doesn't just convey words and their meanings; it also carries other information:

- **Paralinguistic Information**: This includes emotions, attitudes, or feelings of the speaker.
- **Extralinguistic Information**: This tells us about the speaker's identity, such as their age, gender, or health condition.
- **Linguistic Information**: This is about the actual message, including language, dialect, or accent.

There's also something called the "transmittal dimension," which can give clues about where the speaker is physically located.

Given how much information is packed into speech and how easy it is to record, there's growing interest in developing systems that can automatically analyze specific aspects of speech. For example, systems can now identify a speaker's identity, age, or emotions just by analyzing their voice.

One important application of these systems is in healthcare, particularly in diagnosing voice disorders. Traditionally, doctors rely on both instruments and their own judgment to diagnose these conditions. However, the process can be subjective and vary from one doctor to another.

To improve accuracy and objectivity, a new field called **Automatic Voice Condition Analysis (AVCA)** has emerged. AVCA systems analyze voice recordings to identify and classify voice disorders. These systems are efficient, reduce the need for invasive procedures, and can be more cost-effective.

Understanding Voice Disorders

Voice disorders, also known as speech disorders, occur when there is a problem with the way speech sounds are produced, the fluency (flow) of speech, or the quality of the voice.

What is a "Normal" Voice?

Defining a "normal" voice is tricky because what's considered normal can vary depending on who is listening. For instance, a singer might have a unique voice that some people find unusual but is perfectly normal for them. Similarly, a high-pitched voice is normal for a child but might be seen as abnormal in an adult. However, there are some general characteristics of a normal voice:

1. **Pleasant Sound**: The voice should be clear without noise or strange breaks.
2. **Appropriate Pitch**: The pitch (how high or low the voice sounds) should match the speaker's age and gender.
3. **Proper Loudness**: The voice should be loud enough to be heard but not too loud for the situation.
4. **Expressiveness**: The voice should be able to vary in pitch and loudness to convey emotions or meanings.
5. **Sustainability**: The voice should be strong enough to meet the speaker's social and work needs.

Types of Voice Disorders

When a voice doesn't have these normal characteristics, it can be considered abnormal. Three main types of abnormal voices are:

1. **Aphonia**: This is when there is no vibration of the vocal cords, making the voice sound extremely breathy.
2. **Muteness**: This is the complete inability to produce sound.
3. **Dysphonia**: This involves problems with voice quality, pitch, or loudness that don't match the person's age or gender.

Focus of Automatic Voice Condition Analysis (AVCA) Systems

AVCA systems are designed to analyze voices automatically to detect and assess voice disorders. These systems usually don't focus on muteness (since there's no sound to analyze) or aphonia (because it's easy to detect without special systems). Instead, they mainly study dysphonic (abnormal) and normophonic (normal) voices.

Key Voice Traits Analyzed

AVCA systems typically look at two main aspects of the voice:

1. **Vocal Aspects**: This includes loudness (how loud or soft the voice is), pitch (the frequency of the voice), and how these vary in different situations. Problems here might indicate issues like overly aggressive behavior or conditions like Parkinson's disease.

2. **Vocal Quality**: This refers to how the vocal cords vibrate and how the sound resonates in the vocal tract. Issues here could include:
    - **Strain**: Excessive tension in the vocal cords.
    - **Breathiness**: Air leaks because the vocal cords don't close properly.
    - **Roughness**: Irregular vibrations of the vocal cords.
    - **Resonance Problems**: Issues with how the sound is amplified in the vocal tract.

Voice Signal Patterns

Voice signals can be categorized based on their vibrational patterns:

1. **Type I**: Nearly regular and smooth, typical of normal voices.
2. **Type II**: Shows some irregularities, often found in slightly disordered voices.
3. **Type III**: Highly irregular, typical of more severe voice disorders.

In summary, AVCA systems analyze voice disorders by focusing on these vocal aspects and quality. They help in identifying and classifying voice disorders by looking at how the voice deviates from what's considered normal.

How Automatic Voice Condition Analysis (AVCA) Systems Work

AVCA systems are designed to analyze and classify voice conditions, especially to detect voice disorders. These systems generally follow a pattern recognition process, where they extract certain features from a voice recording and then use those features to make decisions about the voice's condition.

Key Steps in AVCA Systems

Before we dive into the details, two main questions need to be addressed:

1. **What type of speech should be analyzed?**
2. **What decisions should the system make based on the analysis?**

1. The Type of Speech to Analyze

The type of speech used in AVCA systems is crucial because different speech tasks can highlight different aspects of voice disorders. Two main types of speech tasks are used:

- **Sustained Phonation**: This is when a person holds a vowel sound for a few seconds. It's a common method in AVCA systems because:
    - It's easy for the system to analyze.
    - The vowel sounds are straightforward to produce.
    - Vowels aren't influenced by things like speech rate, dialect, or emotion.
    - They provide clear, consistent acoustic data.

- Certain vowels, like "a," are preferred because they allow a full examination of the vocal tract. Other vowels, like "i" or "u," might not be as useful because of how the mouth shapes these sounds.
- **Running (Connected) Speech**: This involves normal speaking, like pronouncing words or sentences. Although it's less common in AVCA systems, running speech has its benefits:
    - It reflects more natural speech patterns.
    - It includes dynamic aspects of speech, like the way sounds influence each other.
    - It captures fluctuations in voice quality, which might not appear in sustained vowels.
    - Some voice disorders are more noticeable in running speech than in sustained vowels, making it a valuable tool for detecting certain conditions.

2. The Types of Decisions AVCA Systems Make

AVCA systems can perform three main tasks:

1. **Voice Pathology Detection**: This is a simple decision process where the system decides if a voice is normal (normophonic) or disordered (dysphonic or aphonic). It's like a yes/no decision.
2. **Voice Pathology Identification**: This is more complex because it involves categorizing the type of voice disorder. For example, the system might identify specific conditions like vocal nodules or Reinke’s edema. This task is harder because there are many possible categories to choose from.
3. **Voice Pathology Assessment**: This task involves grading the severity of the voice disorder. For example, the system might evaluate how "hoarse" a voice sounds, which includes traits like roughness and breathiness. Assessment is challenging because it's based on perceptual scales, which can be subjective and vary between listeners.

Challenges in AVCA Systems

AVCA systems face several challenges:

- **Variability**: Normal voices vary widely, which can make it hard to distinguish between normal and disordered voices.
- **Overlapping Conditions**: Some normal and pathological voice traits overlap, making it difficult to classify them accurately.
- **Complexity**: The relationship between voice disorders and their acoustic features is complex and nonlinear, which complicates analysis and classification.

In summary, AVCA systems use specific speech tasks to analyze voice conditions and make decisions about whether a voice is normal, what kind of disorder it might have, and how severe the disorder is. These systems face challenges due to the variability and complexity of human speech.

Overview of Automatic Voice Condition Analysis (AVCA) Systems

1. Input Speech

To build an AVCA system, you need a collection of voice recordings called a dataset or corpus. This dataset is used to train and test the system. When collecting these recordings, it's important to follow certain guidelines to avoid introducing unnecessary variability (like background noise) and to ensure consistency in recording conditions.

Key recommendations for recording voice data include:

- Using high-quality microphones.
- Keeping the mouth-to-microphone distance consistent (less than 10 cm).
- Recording in quiet rooms with low ambient noise.

The dataset should be large enough to cover all variations within the target group, including differences in age, gender, accent, and smoking habits. Proper management of these datasets is crucial, especially when used in clinical settings, as it helps in tasks like diagnosing voice disorders or evaluating voices from a perceptual standpoint.

Existing Datasets

There are several public and private datasets used for detecting and identifying voice disorders:

- **Public Datasets**: One of the most widely used is the Massachusets Ear and Eye Infirmary (MEEI) dataset, which includes recordings of the vowel /a/ and a sentence from the "Rainbow Passage." Another notable dataset is the Saarbrücken Voice Database (SVD), which contains recordings of different vowels and a standard sentence in German.
- **Private Datasets**: These include the Hospital Príncipe de Asturias (HUPA) corpus, which has recordings from Spanish speakers, and the Arabic Voice Pathology Dataset (AVD). Other large datasets have been collected in hospitals in France and other countries, containing recordings of various voice disorders.

2. Preprocessing

Before analyzing the voice recordings, they often need to be preprocessed because speech is naturally non-stationary, meaning it changes over time. Preprocessing helps break down the speech into manageable parts, making it easier to analyze.

- **Short-Time Analysis**: This involves breaking the speech signal into short, equal-length segments called frames, typically 20 to 40 milliseconds long. This allows each segment to be treated as a stable piece of the speech signal for analysis.
- **Windowing**: To improve the quality of the analysis, the beginning and end of each frame are smoothed using a window function, like the Hamming or Hanning windows.
- **Additional Techniques**: Other preprocessing methods include:

    - **Voice/Unvoiced Detection**: Identifying which parts of the speech involve vocal cord vibration.
    - **Silence Detection**: Removing non-speech parts of the recording.
    - **Inverse Filtering**: Removing the effects of the vocal tract from the speech signal to focus on the sound produced by the vocal cords.
    - **Pre-Emphasis Filtering**: Boosting the high-frequency content of speech, although this technique has mixed results in AVCA systems.

In AVCA systems, it's important not to remove certain types of noise, like the irregularities in vocal cord vibrations, because these are often key indicators of voice disorders.

In summary, AVCA systems rely on well-structured datasets and careful preprocessing of voice recordings to accurately detect and analyze voice disorders. The techniques used are designed to handle the natural variability and complexity of human speech.

4.3. Characterization

In AVCA systems, characterization refers to extracting specific features from the voice recordings that can help distinguish between normal and disordered voices. These features are like measurable traits of the voice that provide insight into whether a voice is healthy or not.

The Goal of Characterization

The main goal is to create a "feature vector," which is a collection of measurements that describe different aspects of the voice. This vector helps the system decide if the voice is normal or pathological. However, identifying features that accurately represent voice disorders is challenging because some irregularities in the voice can occur even in healthy voices.

Different Types of Features

To overcome this challenge, AVCA systems often use a combination of different types of features. By analyzing multiple features together, the system can make more accurate predictions. Below are some of the common types of features used:

4.3.1. Temporal and Acoustic Analysis

This type of analysis looks at how the voice changes over time and its basic sound properties:

- **Fundamental Frequency (f0) and Sound Pressure Level (SPL)**: These measure the pitch and loudness of the voice. Some studies use these measurements to distinguish between normal and disordered voices, especially in conditions like spasmodic dysphonia.
- **Voice Breaks and Noise**: Voice breaks and noise in the voice signal can indicate disorders. Techniques that track these irregularities are used to analyze the voice's vibrational patterns.

- **Energy Measures**: The energy in the voice signal, which is influenced by how close the speaker is to the microphone, can also be an indicator of vocal function.
- **Glottal Signal Analysis**: This involves looking at how the vocal cords open and close during speech, providing insights into the voice's quality.

Perturbation and Fluctuation Analysis

These analyses focus on small, short-term disturbances in the voice:

- **Jitter**: Measures the tiny variations in pitch from one sound cycle to the next. It's widely used in AVCA systems to detect voice disorders.
- **Shimmer**: Measures small changes in loudness between sound cycles. Like jitter, it's commonly used to analyze voice quality.
- **Noise Ratios**: These measure the amount of noise relative to the harmonic (pure tone) parts of the voice. Higher noise levels can indicate a voice disorder.

Fluctuation Analysis

This examines larger, more severe disturbances in the voice, often related to neurological disorders:

- **Tremor**: Refers to low-frequency fluctuations in pitch and loudness, which can be signs of neurological issues like Parkinson's disease.

In summary, the characterization stage in AVCA systems involves extracting and analyzing various features from voice recordings. By combining different types of features, the system can better identify and assess voice disorders. These features include basic sound properties, small disturbances, and more significant fluctuations in the voice signal.

4.3.3. Spectral-Cepstral Analysis

What is Spectral-Cepstral Analysis?

Spectral-Cepstral analysis involves examining the sound spectrum and cepstrum (a way of representing the speech signal) to study voice disorders, particularly to understand vocal quality. These features are very effective in detecting voice issues, and they work well for both sustained vowels (like holding the sound "a") and normal speaking (running speech). They don’t rely on estimating the pitch (fundamental frequency), which makes them versatile.

Types of Features in Spectral-Cepstral Analysis:

1. **Spectral Features:**
    - These are derived directly from the speech spectrum (the range of frequencies in the voice).

    - **Long-Term Average Spectrum (LTAS):** Measures energy distribution across different frequencies, often focusing on the balance between low and high frequencies.
    - **Linear Predictive Coding (LPC):** A technique that predicts the speech signal and breaks it down into components related to the vocal tract. LPC coefficients can be transformed into other forms like LPCC (Cepstral Linear Predictive Coding) or mel-transformed versions that relate more closely to how humans perceive sound.
2. **Filter-Bank Features:**
    - These features use filter banks to split the speech signal into different frequency bands for more detailed analysis.
    - **Mel-Frequency Cepstral Coefficients (MFCC):** Perhaps the most popular in speech analysis, MFCCs capture how humans hear and process speech by focusing on perceptually important frequency bands.
3. **Cepstral Peak Prominence (CPP):**
    - Measures the prominence of harmonics (regular patterns in the sound) relative to noise in the voice signal. This is particularly useful for detecting breathiness in the voice, a common symptom of voice disorders.
4. **Other Features:**
    - **Perceptual Linear Predictive Coding (PLP):** Similar to LPC but adjusted to better match human hearing.
    - **Harmonics-to-Noise Ratio (HNR):** Compares the strength of harmonics to noise, helping to detect irregularities in the voice.

4.3.4. Complexity Analysis

What is Complexity Analysis?

Complexity analysis looks at how "complex" the voice signal is, which can relate to how disordered it might be. Complexity, in this case, refers to meaningful patterns rather than randomness. Since the human voice is produced by a complex system (our vocal cords and vocal tract), this type of analysis is particularly useful for identifying irregularities in voice production.

Types of Complexity Features:

1. **Nonlinear Dynamics Analysis (NDA):**
    - This approach examines how the voice signal behaves over time, focusing on properties like nonlinearity and multiscale variability.
    - **Fractal Dimension and Correlation Dimension:** These metrics help measure the complexity of the voice signal's patterns.
2. **Lyapunov Exponent:**
    - Measures how much small differences in the voice signal grow over time, indicating the stability or instability of the voice.
3. **Entropy Measures:**

    - Entropy measures the unpredictability in the voice signal. Higher entropy often indicates more disorder in the voice.
    - **Approximate Entropy (ApEn):** A specific type of entropy used to analyze the regularity and predictability of the voice signal.

4.3.5. 3-Dimensional Representations

What are 3D Representations?

This approach involves creating a 3D visual representation of the voice signal, which can then be analyzed using image processing techniques. These representations help in detecting complex patterns in the voice signal that might be missed in simpler analyses.

Examples:

- **Modulation Spectra (MS):** Visualizes both the modulation and frequency components of speech, helping to identify pathologies.
- **Mel-Spectrograms:** Another type of 3D representation that shows how sound energy is distributed across time and frequency, often analyzed using texture and pattern recognition techniques.

4.3.6. Other Types of Features

Miscellaneous Features:

There are many other approaches to analyzing voice signals that don't fit neatly into the above categories. For example:

- **Multidimensional Acoustic Voice Quality Index:** A combined metric using several features to assess voice quality.
- **Non-Negative Matrix Factorization:** A technique that breaks down the voice signal into its underlying components for analysis.

In summary, spectral-cepstral analysis, complexity analysis, and 3D representations are powerful tools in AVCA systems, allowing for detailed examination of voice signals to detect and classify voice disorders. Each method brings a unique perspective, from analyzing basic sound properties to exploring the intricate dynamics of voice production.

4.4. Dimensionality Reduction

What is Dimensionality Reduction?

Dimensionality reduction is a process used to simplify a large set of features (characteristics) in data, making it easier to analyze. The goal is to remove unnecessary or redundant features that

don't contribute much to the overall analysis, which can improve the performance of systems that process the data.

There are two main approaches to dimensionality reduction:

1. **Feature Extraction:**
    - This method transforms the original features into a new, smaller set of features while keeping as much useful information as possible.
    - The downside is that the new features might be harder to interpret because they don't directly correspond to the original features.
    - Common techniques include:
        - **Principal Components Analysis (PCA):** Reduces data by finding new features that capture the most variance (differences) in the data.
        - **Linear Discriminant Analysis (LDA):** Finds features that best separate different classes (like normal vs. pathological voices).
2. **Feature Selection:**
    - This method picks a subset of the original features without changing them.
    - There are three types of feature selection methods:
        - **Wrapper Methods:** These select features based on how well they improve a model's performance, though they can be computationally expensive.
        - **Filter Methods:** These use statistical measures to select the most relevant features, and they are faster than wrapper methods.
        - **Embedded Methods:** These integrate feature selection directly into the model training process, balancing between speed and performance.

4.5. Machine Learning and Decision Making

How Do Machine Learning Models Work in AVCA Systems?

In AVCA systems, machine learning models are used to make decisions about the voice data. These models learn from a set of examples (training data) to predict outcomes for new data. The learning process can be:

- **Supervised Learning:** The model learns from labeled data, where each example has a known outcome (like "normal" or "pathological").
- **Unsupervised Learning:** The model tries to find patterns in data without any labeled outcomes (not commonly used in AVCA systems).

Common Machine Learning Models in AVCA:

1. **Support Vector Machines (SVM):**
    - A popular model that finds the best boundary between different classes (like normal vs. disordered voices).
    - Used widely in AVCA systems for its accuracy and effectiveness.

2. **Gaussian Mixture Models (GMM):**
    - A model that assumes data is made up of a mixture of different normal distributions (bell curves).
    - It's great for modeling complex voice data and is often combined with other methods like SVM for better results.
3. **Artificial Neural Networks (ANN) and Deep Neural Networks (DNN):**
    - These models mimic how the human brain works and are used for more complex decision-making tasks, especially when large amounts of data are available.
4. **Other Models:**
    - **Random Forests:** A model that uses multiple decision trees to improve accuracy.
    - **k-Nearest Neighbors (KNN):** A simple model that predicts outcomes based on the closest examples in the data.
    - **Bayes Classifier:** A model based on probability that makes predictions based on prior knowledge of conditions.

In summary, dimensionality reduction helps simplify the data by reducing the number of features, making it easier for machine learning models to analyze and make decisions. These models then learn from data to accurately classify or predict voice conditions.

Evaluating the AVCA System

When creating machine learning systems, like those used in AVCA (Automatic Voice Condition Analysis), it's important to check how well they work. To do this, we use validation techniques that help ensure the results are accurate and reliable. Here's how it works:

1. Splitting the Data

To evaluate a system, we usually split the available data into different parts:

- **Training Data:** Used to teach the system how to recognize patterns (like identifying a voice disorder).
- **Testing Data:** Used to check how well the system learned from the training data.
- **Validation Data:** Sometimes used to fine-tune the system's settings.

2. Common Validation Methods

- **Split Sample Method:**
    - The data is split into two parts: one for training and one for testing.
    - This method is simple but may not always give reliable results, especially if the dataset is small.
- **K-Folds Cross-Validation:**
    - The data is divided into *k* equal parts (folds). The system is trained on *k-1* parts and tested on the remaining part. This is repeated *k* times, and the results are averaged.

    - This method is more reliable because it tests the system on different parts of the data.
- **Leave-One-Out Validation:**
    - Similar to k-folds, but with *k* equal to the total number of data points. The system is tested on one data point at a time while being trained on all others.
    - This method is good for small datasets as it uses nearly all the data for training.
- **Bootstrapping:**
    - Random samples of the training data are taken, sometimes with repetition, and the system is tested on different data points each time. The results are averaged to get the final performance.
    - This method gives a sense of how well the system performs across various random samples.
- **Cross-Dataset Validation:**
    - The system is trained on one dataset and tested on a completely different dataset. This method tests how well the system works in real-world scenarios with more variability.

3. Measuring Performance

- **Accuracy (ACC):**
    - The most common measure, which shows the percentage of correct predictions made by the system.
- **Receiver-Operating Curve (ROC) and Area Under Curve (AUC):**
    - These metrics show how well the system can distinguish between different classes (e.g., normal vs. disordered voices). AUC is especially useful because it's not affected by the decision thresholds and provides a standard measure.
- **Other Metrics:**
    - **Log-Likelihood Ratio, DET Curves, Sensitivity vs. Specificity Plots:** These are additional ways to evaluate the system's performance.
    - **Statistical Tests (Mann-Whitney U-test, t-test):** Used to compare the performance between different systems or datasets.

Applications of AVCA Systems

AVCA systems are used to analyze and diagnose a wide range of voice disorders, including:

- **Laryngeal Pathologies:** Such as nodules, polyps, and larynx cancer.
- **Neurological Disorders:** Like Parkinson's and Alzheimer's disease.
- **Sleep Disorders:** Such as obstructive sleep apnea.
- **Other Conditions:** Including hypernasality, dysphagia (swallowing difficulties), and lupus.

In summary, evaluating AVCA systems involves splitting the data into different parts, testing the system on each part, and using various metrics to measure its accuracy. These systems are applied in diagnosing a wide range of voice-related health issues.

Factors Affecting AVCA Systems

When designing Automatic Voice Condition Analysis (AVCA) systems, it's important to understand the various factors that can introduce errors. These factors stem from the natural variability in speech, which can make it challenging for the system to accurately classify voice conditions.

Types of Variability in AVCA Systems

1. **Intra-Class Variability:**
    - This refers to differences within the same group, such as how a person's voice might change based on their age, gender, mood, or health condition.
    - For example, the way someone speaks can vary depending on their dialect, emotional state, or the effort they put into speaking.
2. **Channel-Dependent Factors:**
    - These are external influences that affect the recording quality, such as the type of microphone used, background noise, or echoes in the room.
    - These factors are especially important in telemedicine, where recording conditions can vary widely.

Key Factors Affecting AVCA Systems

1. Intra-Class Variability

This is the variation within the same class of voices that can affect the accuracy of AVCA systems.

- **Dialects and Accents:**
    - Dialects are variations in a language that can affect pronunciation, grammar, and vocabulary. If an AVCA system doesn't account for these differences, it might mistakenly classify a normal voice as disordered.
    - Accents, which are variations in pronunciation, also introduce variability. For example, a system trained on one accent might struggle to accurately classify voices with a different accent, leading to errors.
- **Vocal Effort:**
    - Vocal effort refers to how much energy or force someone uses when speaking. It can change how the voice sounds and affects various aspects of speech, such as pitch, loudness, and duration of sounds.
    - High vocal effort might make a voice sound strained, while low effort could make it sound weak or breathy. These changes can impact the measurements the AVCA system uses to classify voice conditions.
    - For instance, parameters like jitter (small variations in pitch) and shimmer (small variations in loudness) can vary significantly depending on the level of vocal effort, which can complicate the system's ability to accurately assess the voice.

In summary, factors like dialects, accents, and vocal effort introduce variability in speech that can affect the performance of AVCA systems. Understanding and accounting for these factors is crucial for improving the accuracy and reliability of voice condition analysis.

5.1.3. Emotion

Emotions can significantly impact speech, which is why studying emotional content in speech has become a popular area of research. This is known as "affective computing," where systems are designed to automatically detect, recognize, and even simulate human emotions using speech or facial expressions.

Research shows that emotions can affect how well systems recognize speech, especially in children. For example, systems that try to identify emotions before classifying speech can perform worse if the emotional content is strong. Although we don't know much about how emotions impact AVCA systems, it's likely that emotions could make it harder for these systems to accurately assess voice conditions.

5.1.4. Sex

The sex of the speaker introduces a lot of variability in speech-based systems, making it a major factor to consider when designing these systems. Research shows that knowing the sex of the speaker can improve the accuracy of speech recognition systems.

The differences between male and female voices stem from several factors:

- **Physiological Differences:** Males and females have different laryngeal (voice box) anatomy. For instance, males generally have thicker vocal cords, longer vocal tracts, and larger throat areas, which contribute to their deeper voices.
- **Acoustic Differences:** Female voices tend to be "breathier" and have a higher pitch compared to male voices. This is partly because females often have a small gap (posterior glottal opening) in their vocal cords during speech, which is less common in males. This difference affects various acoustic properties, like the harmonic frequencies in their voices.
- **Glottal Waveform Differences:** The pattern of how the vocal cords vibrate (glottal waveform) also differs between sexes. Female glottal waveforms tend to be more symmetric and smoother, while male waveforms are often more asymmetric with a sharper closure phase.

These differences make it crucial to consider the sex of the speaker when designing AVCA systems.

5.1.5. Age

Age also affects voice characteristics, largely due to hormone-related changes:

- **Puberty:** During puberty, boys experience significant changes in their voice due to the development of the Adam's apple and thicker vocal cords, resulting in a deeper voice. Girls also experience changes, but to a lesser extent.
- **Aging:** As people age, especially men, their voices tend to rise in pitch due to muscle atrophy and other age-related changes in the vocal cords. Women may experience a lowering of pitch due to menopause, though the effects of aging on the voice are generally more pronounced in men.

These age-related changes can influence the accuracy of AVCA systems, though this factor hasn't been extensively studied. However, some research suggests that including age as a factor in AVCA models can slightly improve their performance.

5.2. Channel-Dependent External Influences

These are external factors that affect the quality of voice recordings, adding variability to the data:

- **Recording Equipment:** The type of microphone, the analog-to-digital converter, and other equipment used can influence the quality of the recording. For example, different microphones can change the sound spectrum, which may affect the accuracy of voice analysis.
- **Acoustic Environment:** The place where the recording is made, whether it's a quiet studio or a noisy office, can also impact the recording quality.
- **Transmission Means:** How the voice is transmitted (e.g., over a phone line or via cellular networks) can introduce noise or distortions that complicate analysis.
- **Background Noise:** Any external sounds or noises made by the speaker (like lip smacks) can interfere with the recording and reduce the accuracy of the AVCA system.

Research has shown that factors like microphone sensitivity, distance from the mouth, and background noise levels all play significant roles in the accuracy of speech analysis. For example, recordings with a signal-to-noise ratio below 30 dB can negatively affect the analysis, and certain microphones are better suited for these tasks than others.

In summary, emotions, the speaker's sex, age, and external recording factors are all important considerations in designing effective AVCA systems. Each of these factors introduces variability that can challenge the system's ability to accurately assess voice conditions.

6. Discussion and Conclusions

This paper has explored the concepts related to voice disorders and the systems used to analyze them, known as AVCA (Automatic Voice Condition Analysis) systems. The paper provided an overview of how these systems work and reviewed the most common techniques used in the field.

Here are the key points discussed:

1. **Datasets and Their Limitations:**
    - Many studies still use the MEEI dataset, despite its known limitations. However, newer public datasets like the SVD and private datasets shared among researchers have allowed for better comparisons and more reliable results.
    - There's still a need for larger, more diverse datasets that include a wider variety of speech samples and are balanced in terms of pathology, age, and sex.
2. **Types of Speech Analyzed:**
    - Most studies focus on analyzing sustained vowels because they are simple to work with. However, running speech (normal conversation) has a lot of potential for analysis, and more research is needed in this area.
3. **Impact of Variability Factors:**
    - Factors like age, sex, and accent can affect the accuracy of AVCA systems, but these are often overlooked in research. Future studies should pay more attention to these factors to improve system performance.
4. **Characterization Techniques:**
    - Most studies focus on measuring vocal quality, but other aspects like variations in intensity and pitch (f0) are also important and can help in identifying conditions like low voice volume (hypophonia) or abnormal pitch.
5. **Machine Learning and Decision-Making:**
    - Traditional machine learning methods like k-nearest neighbors (k-NN) and Linear Discriminant Analysis (LDA) are common, but newer methods like Gaussian Mixture Models (GMM) and Support Vector Machines (SVM) are becoming more popular due to their effectiveness.
    - Advanced methods like Universal Background Models (UBM) and deep learning approaches (e.g., deep neural networks) are promising but require large datasets, which are often not available.
6. **Validation and Testing:**
    - There are some methodological issues in existing studies that can affect the reliability of results. For example, using data from the same speaker in both training and testing phases can bias the results. Researchers should also be careful about reporting overly confident accuracy results.
    - It's important to validate AVCA systems in real clinical settings, where they can be tested for their ability to aid in diagnosing voice disorders.

In summary, while there has been significant progress in the development of AVCA systems, there is still room for improvement, especially in the areas of dataset diversity, consideration of variability factors, and real-world testing.

Paper 2

1. Introduction

Voice impairments can be caused by a variety of factors such as misuse, infections, neurological issues, vocal abuse, surgery, trauma, or exposure to harmful substances. Traditionally, detecting these impairments involves both objective (instrumental) and subjective (perceptual) evaluations, along with other medical tests to determine the presence and severity of a voice disorder.

To help doctors in diagnosing these issues, a field called Automatic Voice Condition Analysis (AVCA) has emerged. AVCA offers several benefits over traditional methods, including being more objective and non-invasive since it uses speech signals for analysis.

The paper explores advanced techniques used in speaker recognition systems, particularly those based on Gaussian Mixture Models (GMM). GMMs are statistical models that represent the data using a combination of multiple Gaussian distributions. When there is a lot of training data, GMMs can be very effective. However, when data is limited, other methods are preferred.

One such method involves using a Universal Background Model (UBM), which is a GMM trained on a larger, separate dataset. This UBM is then adapted using the specific training data to create more accurate models known as GMM-UBM. These models are widely used in speaker recognition. Variations of this approach, like GMM-SVM and i-Vectors (IV), are also discussed, which further improve the system's accuracy by handling variability in the training data.

2. Experimental Setup

2.1. Acoustic Material (Datasets)

The experiments use three main datasets that contain recordings of both normal (normophonic) and pathological (voice disorder) speech:

1. **HUPA Dataset:**
    - Recorded at Príncipe de Asturias Hospital in Madrid, Spain.
    - Includes recordings of 366 adult Spanish speakers saying the vowel "a."
    - The recordings are high-quality, with a sampling rate of 50 kHz and 16-bit resolution.
    - The dataset includes a variety of voice disorders, such as nodules, polyps, and carcinomas.
2. **GMar Dataset:**
    - Collected at Gregorio Marañón Hospital in Madrid, Spain.
    - Contains recordings of Spanish speakers saying the vowels "a," "i," and "u."
    - Recorded with a sampling rate of 22.05 kHz and 16-bit resolution.
    - Includes 202 recordings, with a mix of normal and pathological voices.
3. **SVD Dataset:**

- Contains recordings from over 2000 German speakers, both normophonic and pathological.
- Recorded at Saarland University and Caritas Clinic St. Theresia in Germany.
- Includes various recordings like vowels at different pitches and a sentence in German.
- The dataset was filtered to remove low-quality recordings, leaving 1538 recordings for analysis.

2.1.4. Ancillary Datasets

Four additional datasets are used to create a Universal Background Model (UBM), which helps improve the accuracy of the system:

1. **EUROM Corpus:** Contains recordings of 60 speakers in several European languages. Only the German portion is used in this study.
2. **Albayzin Dataset:** A Spanish dataset designed for speech recognition, including 6800 recordings of balanced phrases.
3. **PhoneDat-I Dataset:** Contains about 20,000 recordings of German speakers reading sentences and fables.
4. **MEEI Voice Disorders Dataset:** A well-known dataset with 710 recordings of English speakers with various voice disorders, including both vowel sounds and reading passages.

2.1.5. Cross-Dataset Trials

Two additional datasets are used to test the system's performance across different datasets:

1. **DN Partition:** Recorded at Hospital Dr. Negrín in Las Palmas de Gran Canaria, Spain, with 181 recordings of the vowel "a" from Spanish speakers.
2. **ATIC Dataset:** Includes recordings from 79 Spanish speakers (both dysphonic and normophonic) recorded in quiet rooms under controlled conditions.

These datasets and experimental setups are used to evaluate how different factors affect the performance of AVCA systems, aiming to create a system that can perform well across various conditions and datasets.

2. Methodological Setup

This section explains the setup for four main experiments conducted in the study. The goal is to examine how different factors affect the performance of AVCA (Automatic Voice Condition Analysis) systems and to test various classification techniques commonly used in speaker recognition.

2.2.1. Experiment 1: Impact of Acoustic Material and Feature Sets

This experiment tests how different types of speech data and feature sets influence the performance of AVCA systems. Since no single feature can fully differentiate between healthy and impaired voices, this experiment helps identify the best combination of features.

- **Setup:** The experiment uses acoustic data from the HUPA, SVD, and GMar datasets. Different feature sets, such as vocal quality descriptors, are extracted from the speech data.
- **Process:** The data is preprocessed, features are extracted, and then a decision-making process (using GMM classifiers) is applied. The effectiveness of the features is evaluated using metrics like the Area Under the ROC Curve (AUC) and others.

2.2.2. Experiment 2: Role of Speaker's Sex

This experiment explores whether creating separate AVCA systems for male and female speakers improves performance. The idea is that by tailoring the system to specific speaker characteristics, it may be more accurate.

- **Setup:** Three trials are conducted using the vowel "a" from the HUPA, SVD, and GMar datasets. Data is split by the speaker's sex, and features like MFCC (Mel-Frequency Cepstral Coefficients) are extracted and analyzed.
- **Process:** Similar preprocessing, feature extraction, and classification steps are used as in Experiment 1, but with a focus on sex-specific models.

2.2.3. Experiment 3: Testing Classification Techniques from Speaker Recognition

This experiment tests how well advanced classification techniques from speaker recognition (like GMM-UBM and i-Vectors) work for AVCA systems.

- **Setup:** The experiment uses recordings of the vowel "a" and running speech from the SVD dataset. It tests different configurations of training data and classification techniques.
- **Process:** Data is preprocessed, features (MFCC) are extracted, and various classifiers are applied. The results are then evaluated using the same metrics as in previous experiments.

2.2.4. Experiment 4: Combining the Best Systems

This experiment builds on the insights from the previous experiments to create a comprehensive AVCA system. It aims to combine the most effective features and classifiers to provide a single, reliable decision about a speaker's voice condition.

- **Setup:** Two trials are conducted: one combines results from different speech tasks (like vowels and running speech), and the other tests the system across different datasets.
- **Process:** Data is preprocessed, the best features from Experiment 1 are selected, and the system is trained using GMM classifiers. Results from different systems are combined using logistic regression to improve accuracy.

Each of these experiments is carefully designed to test different aspects of AVCA systems, with the ultimate goal of improving their accuracy and reliability in diagnosing voice conditions.

3. Results

3.1. Experiment 1: Impact of Acoustic Material and Feature Sets

This experiment tested 9 different trials using various vowels (/a/, /i/, /u/) and running speech from the HUPA, GMar, and SVD datasets to see how different types of speech and features affect the performance of AVCA systems.

3.1.1. HUPA Dataset

- **Results:** The best performance came from the Perturbation (Pert) feature set with an AUC (Area Under the Curve) of 0.85. The Spectral/Cepstral (SCs) set, including MFCC and PLP features, also performed well with AUCs around 0.79-0.80. The Complexity (Comp) features had mixed results, with the Entropy (Ent) subset doing well (AUC of 0.83) and the Long-Range (LR) subset performing poorly (AUC of 0.60).

3.1.2. GMar Dataset

- **Results:** For the vowel /a/, the Entropy subset of Comp features had the best performance (AUC = 0.83). For the vowel /i/, the Modulation Spectrum (MSs) features performed best (AUC = 0.76). For the vowel /u/, the Long-Range (LR) and Entropy subsets performed similarly well (AUCs around 0.73-0.74).

3.1.3. SVD Dataset

- **Results:**
    - **Vowel /a/:** The Perturbation features again performed best (AUC = 0.78), followed by MFCC and PLP (AUCs around 0.76-0.77).
    - **Vowel /i/:** MFCC and PLP features were the best (AUC = 0.75-0.76), with Pert and MSs also doing fairly well.
    - **Vowel /u/:** MFCC and PLP were again top performers.
- **Running Speech:**
    - Using raw speech, MFCC and PLP features performed equally well (AUCs around 0.85-0.86).
    - For vowels extracted from running speech, MFCC features were slightly better (AUC = 0.86) compared to PLP (AUC = 0.85).

3.1.4. General Observations

- **Feature Performance:** No single feature set was consistently the best across all conditions. However, Pert and SCs sets often showed strong performance across different datasets and speech tasks.

- **Acoustic Material:** The vowel /a/ generally performed better than /i/ and /u/. When using running speech, spectral/cepstral features like MFCC and PLP tended to perform better than when using just sustained vowels.
- **Challenges:** Comparing results between different speech tasks (like running speech vs. sustained vowels) is tricky because of differences in the amount and type of data available. For instance, running speech involves more frames to analyze, which can impact the results.
- **Conclusion:** The results suggest that using running speech, especially without segmenting out voiced parts, may simplify the AVCA system and still provide strong performance. This comes at the cost of processing more data, but it avoids potential errors from automatic segmentation processes.

3.2. Experiment 2: Effects of Extralinguistic Traits

Figure 7 shows the performance metrics for systems that account for the sex of the speaker (sex-dependent) versus those that do not (sex-independent) across different datasets.

- **HUPA Dataset:** Including the sex of the speaker improves performance, with the sex-dependent model achieving an AUC of 0.81 compared to 0.79 for the sex-independent model. Specifically, separating models for females and males yields higher AUCs (0.85 for females, 0.81 for males).
- **SVD Dataset:** Similarly, considering the sex of the speaker leads to a slight improvement (AUC = 0.78 vs. 0.77). Interestingly, the female-specific model performs better (AUC = 0.79) than the male-specific model (AUC = 0.77).
- **GMar Dataset:** The trend continues, with the sex-dependent model outperforming the sex-independent one (AUC = 0.78 vs. 0.77). Here, the male-specific model shows better performance (AUC = 0.82) compared to the female-specific model (AUC = 0.77).

Key Observations:

- **Sex Information:** Accounting for the speaker's sex generally improves the system's efficiency in detecting voice pathologies across all datasets. Performance improvements ranged from 4% (HUPA) to 0.3% (SVD).
- **MFCC Variability:** The number of MFCC coefficients that optimize performance differs between male and female models, likely due to physiological differences in vocal tracts and folds.
- **Female vs. Male Models:** Systems trained on female voices often perform worse than those trained on male voices, possibly due to pitch differences and the presence of a glottal gap in female speakers. This pattern has been observed in other speech-related applications.

3.3. Experiment 3: Testing Classification Techniques from Speaker Recognition

This experiment evaluates the effectiveness of classification techniques commonly used in speaker recognition for detecting pathological speech. Two trials were conducted using sustained phonation and running speech from the SVD dataset, with different configurations for training the UBM-based classifiers.

3.3.1. Trial with Sustained Phonation

Three configurations were tested:

1. **C1:** Used a small amount of normophonic data.
2. **C2:** Increased the amount of normophonic data.
3. **C3:** Combined normophonic and dysphonic data.

Results:

- The GMM-UBM model did not improve much over the baseline GMM. However, more complex models like PLDA showed slight improvements, achieving the best AUC of 0.80.
- Using only normophonic data for training the UBM and compensation models led to better performance, particularly as more normophonic data was added.

3.3.2. Trial with Running Speech

Five configurations were tested using different combinations of normophonic and dysphonic vowels, voiced segments, and sentences from various ancillary datasets.

Results:

- The simple GMM classifier gave the best overall results.
- The best configuration (C3) used normophonic sentences, suggesting that adding sustained phonation data doesn't improve performance.
- Configuration C1, which used both normophonic and dysphonic registers, performed the worst, indicating that mixing these registers for training might reduce effectiveness.

3.4. Experiment 4: Combining the Best Systems

This experiment aimed to design an AVCA system using the insights from previous experiments, focusing on the best feature sets, hierarchical models based on speaker sex, and speaker recognition techniques. The approach involved selecting and ranking features for consistency and generalizability across datasets.

Key Features:

- **GNE, CHNR, and RALA** emerged as the most consistent features across different datasets and speech tasks.

System Design:

- An AVCA system was built using these top features and GMM classifiers, combined with hierarchical categorization based on speaker sex.
- The system's decision scores were fused with results from UBM-based classifiers to enhance performance.

3.4.1. Intra-Dataset Trial

Testing on the SVD dataset using vowels and running speech showed that combining results from different speech tasks improved performance, achieving an AUC of 0.88.

3.4.2. Cross-Dataset Trial

The system, trained on the SVD dataset, was tested on different datasets (GMar, HUPA, ATIC, DN).

Results:

- The system performed well, with AUCs ranging from 0.75 to 0.94.
- The highest AUCs were achieved when using the vowel /a/, indicating its robustness across datasets. However, no additional improvement was observed when combining different vowels for the GMar corpus.

4. Discussion

The results show that dividing data by the speaker's sex improves the performance of automatic voice condition analysis (AVCA) systems, with improvements ranging from 0.3% to 4% depending on the dataset. This aligns with previous research that found slight performance boosts when datasets are manually segmented by sex. This improvement suggests that breaking down the voice pathology detection task into smaller sub-problems based on sex can enhance system performance. This is likely because male and female vocal tracts and folds differ, leading to different spectral characteristics that impact AVCA systems. By addressing these differences separately, the detection process becomes simpler and more effective.

When it comes to the features used in detection, the results confirm that no single measurement can fully capture the complexities of dysphonia. This suggests that multidimensional approaches, combining multiple features, are more effective for AVCA systems. The most consistent and effective features identified were GNE, RALA, and CHNR, with CPPS and PE also showing strong performance. These features come from different contexts but work well together, demonstrating their complementary nature.

In Experiment 3, which tested speaker recognition classification techniques, the classifiers performed well with sustained phonation when paired with normophonic data. However, performance did not improve when using running speech, likely due to differences in language and phonetic content between the training and test datasets. This mismatch highlights the

importance of using training data that closely matches the target dataset, especially in terms of language and phonetic content. The better results with sustained phonation might be because it is less affected by language differences, and normophonic data likely has a more compact representation in the feature space, making it easier to model.

The findings also underscore the need for larger, more balanced datasets that consider factors like age and sex, as these can significantly affect AVCA systems. There's room for further improvement by exploring other types of features and classifiers that are better suited to aging voices or sex differences. Additionally, future research should investigate other factors that influence performance, such as accents and vocal effort. It might also be beneficial to incorporate extralinguistic information, like age or sex, into the system more effectively, possibly through a-priori probabilities or regression models.

5. Conclusions

The current research in Automatic Voice Condition Analysis (AVCA) systems often shows promising results in controlled environments, suggesting that the problem of voice pathology detection is nearly solved. However, these systems are still not ready for clinical use because their high accuracy is mostly achieved under laboratory conditions and depends heavily on the specific dataset used.

This paper has established a more realistic baseline for AVCA systems by considering various factors that affect system robustness. The study analyzed the impact of different speech tasks, extralinguistic factors (like sex), acoustic features, and classifiers on the performance of AVCA systems. The methodology used aimed to produce results that are not dependent on a single dataset, making them applicable to other datasets as well.

Key findings include:

1. **Importance of Extralinguistic Factors**: The study showed that accounting for the speaker's sex improves system performance by 0.3% to 4%, depending on the dataset. This suggests that breaking down the voice pathology detection task based on extralinguistic factors like sex simplifies the problem and enhances system efficiency.
2. **Multidimensional Approaches**: The research confirmed that no single feature can fully capture the complexity of vocal pathology. Therefore, multidimensional approaches, which combine multiple features, are necessary. The most consistent features identified were GNE, CHNR, and RALA, which, when used together, provided better detection results.
3. **Speaker Recognition Techniques**: The study tested speaker recognition classifiers, showing that these techniques work well when sustained vowels are used along with normophonic data. However, when running speech was used, the results were less clear, likely due to mismatches between the training and testing datasets in terms of language and phonetic content.
4. **Cross-Dataset Robustness**: The system demonstrated robustness in cross-dataset scenarios, achieving an Area Under the Curve (AUC) between 0.75 and 0.94, which

indicates that the methodology can handle mismatches between training and testing conditions effectively.

In conclusion, the study suggests that a methodology based on hierarchical detection, the use of a reduced set of consistent features, and the fusion of different speech tasks can significantly enhance the performance of AVCA systems. Future work will focus on testing these systems in clinical settings and exploring the impact of other extralinguistic factors like age, mood, and accent to further improve accuracy and applicability in real-world scenarios.

Everything is not officially done. There are mistakes. Everything written here is temporary.

The MATLAB script called **Process.m** is designed to process a directory of audio files in order to extract features that can be used for machine learning or signal analysis.

The folder path used is →
`"PC-GITA_per_task_44100Hz\PC-GITA_per_task_44100Hz\sentences\laura\sin normalizar\HC"`

1. Input Directory and Initialization

The script begins by specifying the **audio directory**, defined in the variable `audioDir`, where all the `.wav` files are located. The goal is to process these `.wav` files to extract consistent features across them. It standardizes the **sampling frequency** to 16,000 Hz (`target_fs`) so that all files have the same baseline frequency for analysis. The script also sets up two storage structures: `allFeatures`, a cell array used to store the feature matrices of each audio file, and `allLabels`, which holds labels corresponding to each audio file.

2. Audio File Processing

The script processes every `.wav` file in the directory one by one. Each audio file is first **loaded** using `audioread`. If the audio has two channels (stereo), it is converted to mono by averaging these channels together. The script then **resamples** the audio to 16,000 Hz to make sure every file uses the same frequency. For very short audio files (those with fewer than 256 samples), the script decides to **skip processing** as they don't contain enough data for meaningful analysis.

3. Denoising Using Wavelets

To improve the quality of the audio signals, the script **denoises** each audio file. This is done using wavelet decomposition (`wavedec`). By applying **universal thresholding** to the wavelet coefficients, noise can be effectively removed, and the denoised signal is reconstructed using `waverec`.

4. Feature Extraction

The script extracts two key types of features from the audio files: **MFCCs** (Mel-Frequency Cepstral Coefficients) and **pitch**. Specifically, it calculates 13 MFCCs to capture the spectral properties of the audio. If an audio file is too short for MFCC extraction (less than 512 samples), it is skipped. Additionally, the **pitch** is estimated using the `pitch` function, with a frequency range between 50 and 400 Hz. Sometimes, errors may occur during pitch extraction, and files

with such errors are also skipped. Since MFCC and pitch features may not always align, the pitch values are **interpolated** to ensure they match the frame count of the MFCCs.

5. Feature Concatenation

Once MFCCs and pitch features are successfully extracted, they are **combined** into a single feature matrix for each audio file. If the feature extraction for any file fails, that file is skipped entirely.

6. Label Assignment

The script assigns **placeholder labels** to the processed files for demonstration purposes. Each label is simply the index of the loop (`i`), but these labels can be replaced by real labels when working on actual supervised learning tasks.

7. Error Handling

The script is designed to handle errors gracefully. It will skip over files that cause problems, such as being too short or having issues during feature extraction. This ensures that the entire process runs smoothly without major interruptions, and relevant **warning messages** are provided.

8. Feature Normalization for Model Input

Since audio files can have different feature lengths, the script pads the feature matrices so they are all the same length. This involves creating a **3D matrix**, `paddedFeatures`, where each feature matrix is padded with zeros to match the maximum length found among the files.

9. Summary

The final output is a 3D array called `paddedFeatures`. In this array:

- The first dimension represents the **file index**.
- The second dimension represents the **time/frame axis**, padded to the longest file.
- The third dimension contains the **feature values** (MFCC + pitch).

This processed dataset can then be used for various applications, such as training machine learning models for tasks like audio classification or speaker recognition. It can also be used for statistical analysis or to benchmark the performance of different feature extraction and preprocessing strategies.

This script highlights the importance of handling **variable-length audio data**, using denoising techniques, and extracting useful features. It ensures reproducibility by standardizing sampling

rates and aligns different types of features for consistent processing. It also includes robust error handling to prevent interruptions during the processing of large audio datasets.

The **Feature_extraction.m** script is used for **visualizing** the features extracted from a specific audio file. Let me explain how this visualization works:

1. File Selection

The script allows you to select a specific file by setting an index (`fileIndex = 10`). It then checks to make sure that the chosen index is within the available number of files. If the index is out of range, an error message appears to prevent runtime problems.

2. Feature Extraction

Once the file is selected, the corresponding feature matrix (`selectedFeatures`) is retrieved from `allFeatures`. This feature matrix is split into **MFCC features** (the first 13 columns) and **pitch features** (the following column).

3. Visualization

The visualization is split into two parts:

3.1 MFCC Features

The MFCC features are visualized using a **heatmap**. Each column of the heatmap represents a frame in time, while each row represents an individual MFCC coefficient. The x-axis shows the time progression, and the y-axis represents the different spectral features of the audio. A **colorbar** is added to provide a visual scale of the MFCC values.

3.2 Pitch Features

The **pitch contour** is plotted using a line graph, with the x-axis representing the frame index over time, and the y-axis showing the pitch value in Hertz (Hz). This visualization allows you to see how the fundamental frequency changes throughout the audio file, which is helpful for analyzing **intonation, prosody**, and other audio characteristics.

4. Purpose

These visualizations serve a few different purposes:

- The **MFCC heatmap** shows how the spectral features of the audio evolve over time, which is helpful for understanding the audio content.

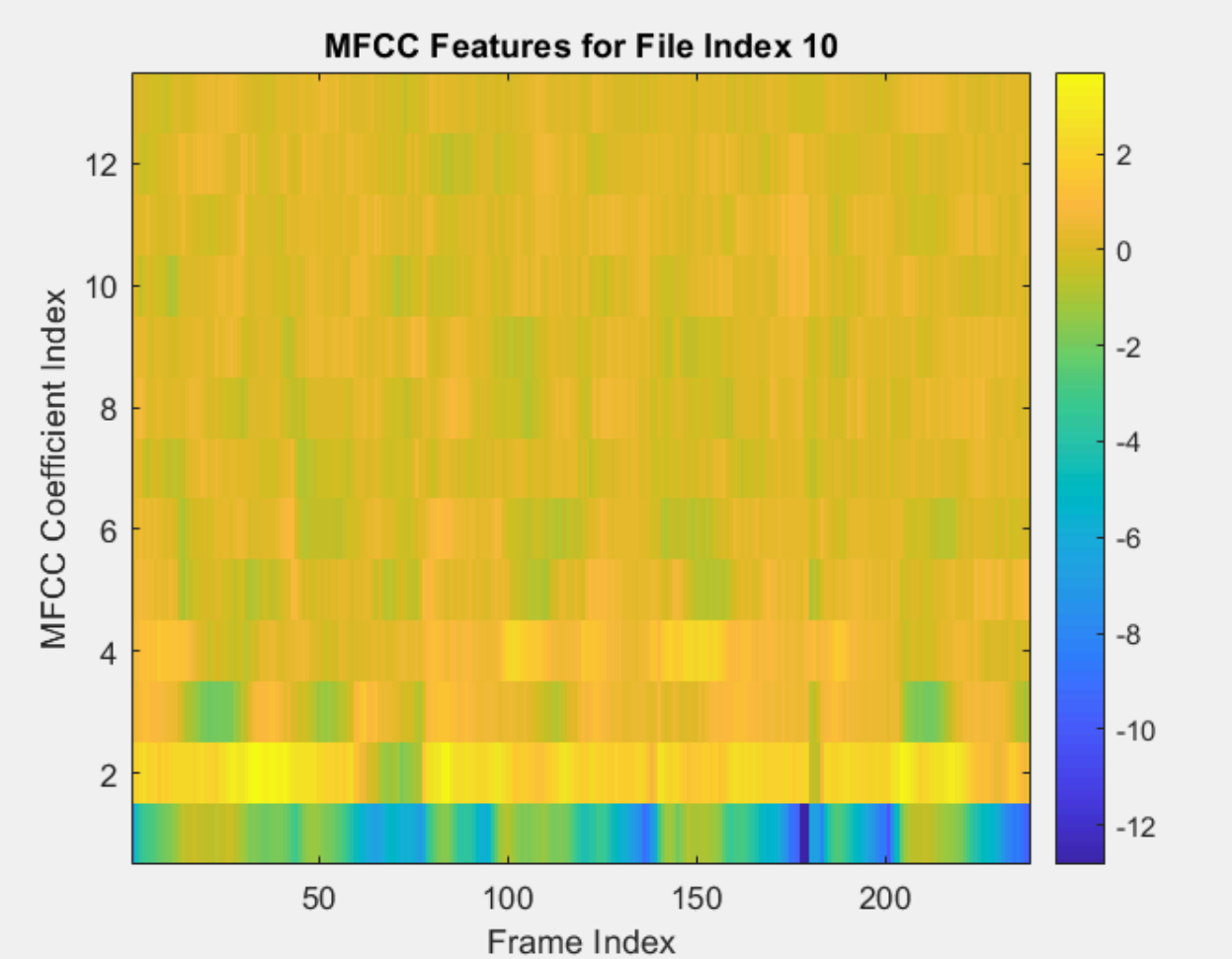
- The **pitch contour** helps analyze the pitch variations, which is particularly useful in tasks like speech analysis.
- Together, these visualizations are great for **exploratory data analysis**, debugging feature extraction processes, and presenting the results of feature extraction.

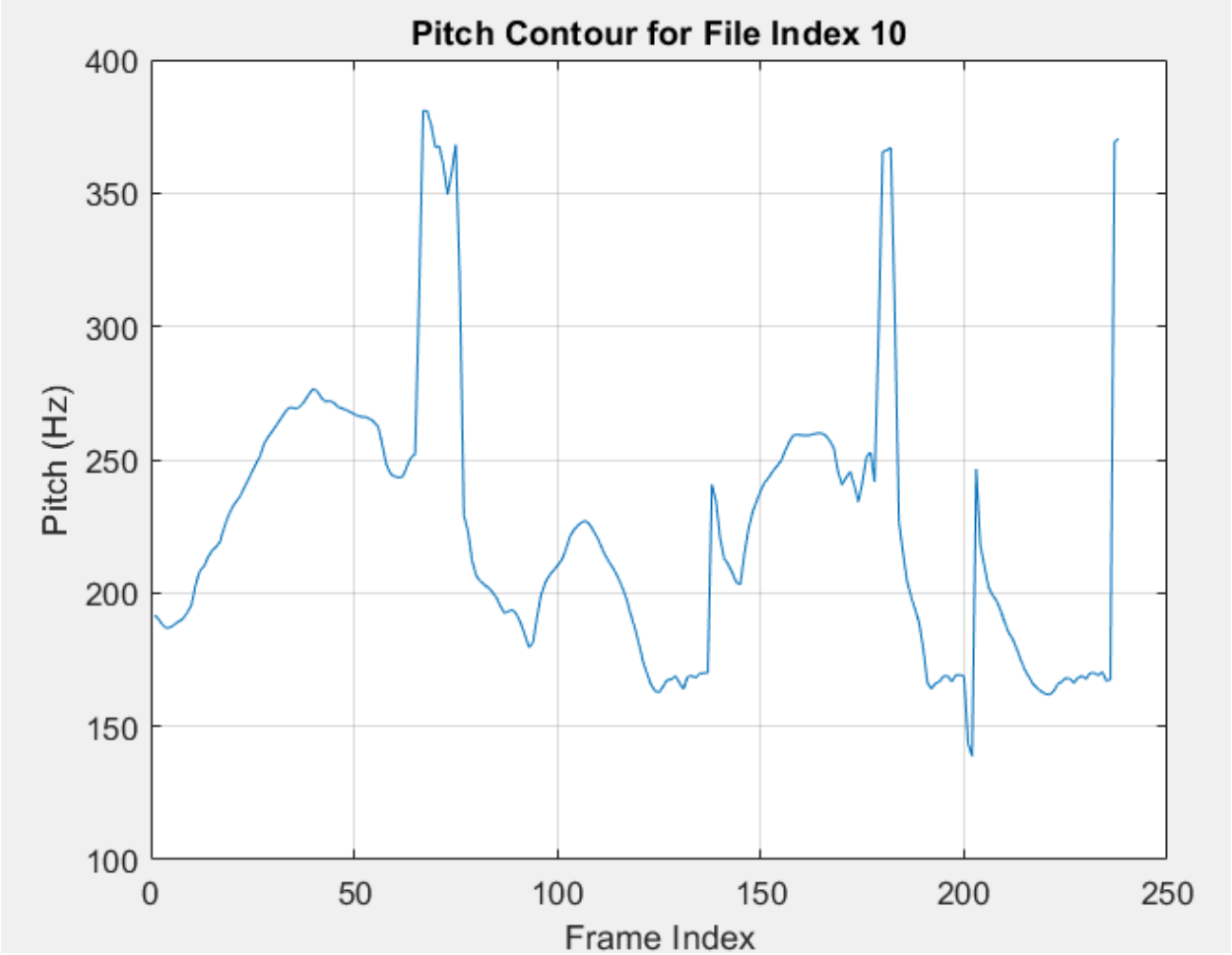
Output

The script generates two figures:

1. A **heatmap** of MFCC features.
2. A **line plot** of the pitch contour.

These visualizations help provide a comprehensive understanding of the features extracted from the audio files, making them valuable for understanding the dataset and guiding further processing steps.





The MATLAB script **Time_spectral.m** is designed to calculate and visualize the average features (such as MFCC coefficients and pitch) across all audio files in a dataset. This provides an overview of the dataset's spectral and pitch characteristics, which can be helpful for statistical analysis or as input for training machine learning models.

1. Initialization

The script begins by setting up **accumulators** to store the total values of each feature:

- `totalMFCC`: This is a vector initialized to zero, which will be used to accumulate the average MFCC values from each audio file.
- `totalPitch`: A scalar initialized to zero, which will sum the average pitch values from each file.
- `count`: A variable that keeps track of the number of processed files, which is used for normalization later.

2. Loop Through All Files

The script then loops through each file in `allFeatures`:

- **Feature Extraction**: For each file, the script separates the **MFCC coefficients** (`mfccs`) and **pitch values** (`pitch`) from the feature matrix.
- **Mean Calculation**: The mean MFCC coefficients are calculated using `mean(mfccs, 1)`, which computes the average of each coefficient across all frames for the current file. Similarly, the mean pitch value is calculated using `mean(pitch)`.
- **Update Accumulators**: The `totalMFCC` vector is updated by adding the mean MFCC coefficients of the current file, and `totalPitch` is updated by adding the mean pitch value.
- **File Count**: The `count` variable is incremented to keep track of how many files have been processed.

3. Average Feature Calculation

Once all the files have been processed:

- The **average MFCC** (`avgMFCC`) is calculated by dividing the `totalMFCC` by the `count`.
- The **average pitch** (`avgPitch`) is similarly calculated by dividing `totalPitch` by the `count`.

4. Visualization

The script then visualizes the calculated average features:

4.1. Average MFCC Coefficients

- A **bar plot** is created to display the average values of each MFCC coefficient across all audio files.
- **Axes and Title**:
    - The **x-axis** represents the MFCC coefficient index (from 1 to 13).
    - The **y-axis** shows the average value of each coefficient.
    - The **title** indicates that the plot represents the average MFCC coefficients.

4.2. Average Pitch

- The average pitch value (`avgPitch`) is displayed in the **command window** with a descriptive message such as:
  Average Pitch Across All Audio Files: 125.34 Hz

5. Purpose

- **Dataset Overview**: The average MFCC coefficients provide a compact summary of the dataset's spectral characteristics, while the average pitch gives a representative measure of the fundamental frequency range present in the dataset.
- **Data Validation**: This process also helps identify any inconsistencies in the dataset, such as unusually high or low values that might indicate errors in data preprocessing.
- **Feature Engineering**: Understanding the trends and patterns within the dataset can guide further analysis or inform the selection of features for training machine learning models.

6. Applications

- **Speaker Analysis**: These average features can help analyze spectral and pitch characteristics for a group of speakers or audio samples.
- **Speech Recognition**: The average spectral and pitch features can inform decisions about normalization or scaling strategies for features used in speech recognition models.
- **Quality Assurance**: This analysis ensures that the feature extraction process yields reasonable and consistent results across all files.

Output

- The script produces a **bar chart** that shows the average MFCC coefficients.
- It also prints a message showing the **average pitch value** in Hertz (Hz), which helps provide both visual and numerical insights into the dataset's audio characteristics.

The **CNN.m** MATLAB script processes a directory of audio files, converts them into spectrogram representations suitable for deep learning, and trains a **Convolutional Neural Network (CNN)** to classify the audio data into two categories. Here is a detailed breakdown of what the script does:

1. Input Data Preparation

- The script begins by specifying the **directory path** (`audioDir`) where the audio files are stored. All `.wav` files in the directory are retrieved using the `dir` function, ensuring that the code processes all audio files available in the directory.
- To standardize the dataset, all audio files are **resampled** to a target sampling frequency of **16,000 Hz** (`target_fs`).

2. Audio Processing Loop

The script then processes each audio file within a loop to generate **spectrograms** for training the CNN:

- **Loading Audio**: Each audio file is loaded using `audioread`, and if the audio is stereo, it is converted to mono by averaging the channels.
- **Resampling**: The audio is then resampled to the target frequency to ensure consistency.
- **Spectrogram Calculation**: The script calculates the spectrogram using specific parameters:
    - **Window Size**: `256` samples for the Short-Time Fourier Transform (STFT).
    - **Overlap**: An overlap of **80%** between consecutive windows is used for smoother transitions.
    - **FFT Points**: `nfft = 512` to determine the frequency resolution.
    - The **spectrogram** is computed and its magnitude is converted to a **log scale** to prevent issues with taking the logarithm of zero.
- **Normalization and Resizing**: Spectrogram values are normalized to fall between **0** and **1** and are resized to **128 x 128** pixels to match the input size required by the CNN.
- **Dataset Accumulation**: The generated spectrograms are stored in a **4D array**, where each spectrogram represents a grayscale image of size 128x128. **Labels** are also generated, although in this example they are randomly assigned as `0` or `1`.

3. Labels Conversion

- Labels are converted into **categorical format** using the `categorical` function to prepare them for training a classification model in MATLAB.

4. CNN Architecture

The script defines a simple CNN for **binary classification**:

- **Input Layer**: Accepts 128x128 grayscale images.
- **Convolutional Layers**: Two convolutional layers, each with `3x3` filters:
    - The first layer has **16 filters** and the second has **32 filters**.
    - **Padding** is used to ensure the output size matches the input size.
    - **ReLU Activation** is applied for non-linear transformations.
    - **Batch Normalization** is used to stabilize and accelerate training.
- **Pooling Layer**: A max pooling layer reduces the spatial dimensions with a stride of 2.
- **Fully Connected Layer**: Maps features to two output nodes for binary classification.
- **Softmax Layer** and **Classification Layer**: Used for computing classification probabilities and loss.

5. Training Options

The training options are specified using `trainingOptions`:

- **Optimizer**: The **Adam** optimizer is used with an initial learning rate of **0.001**.
- **Epochs**: The network is trained for **10 epochs**.

- **Mini-Batch Size**: The mini-batch size is set to **32** samples.
- **Visualization**: Training progress is displayed, showing metrics like loss and accuracy.

6. CNN Training

The `trainNetwork` function is used to train the CNN:

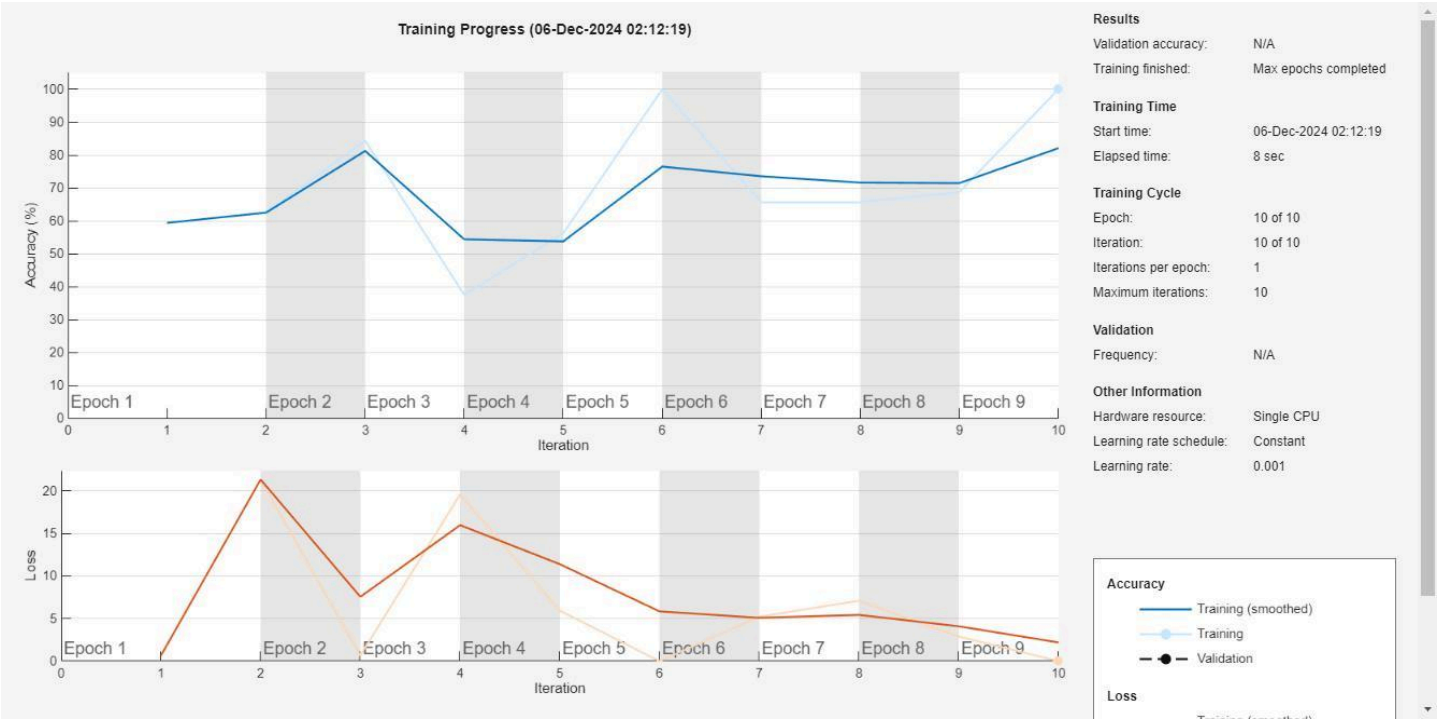
- Inputs include the spectrograms, labels, network architecture, and training configurations.
- The output is a trained network (`net`) that can classify the spectrograms into one of the two categories.

Purpose of Each Step

- **Audio Preprocessing**: Converts the raw audio into spectrograms, which serve as input to the CNN.
- **Normalization and Resizing**: Ensures compatibility with the CNN and improves the stability of training.
- **CNN Architecture**: Uses convolutional layers to extract features from the spectrograms and classify the audio data.
- **Training Options**: Configures the optimization process to ensure efficient training.

Final Output

- The output is a trained CNN (`net`) that can classify audio spectrograms into two categories. During training, the progress is displayed, showing metrics like **loss** and **accuracy** over the epochs.



The image shows the training progress of a Convolutional Neural Network (CNN) over 10 epochs, and here's a detailed breakdown of what's happening:

1. **Accuracy Plot (Top Panel)**
   Observation: The accuracy increases steadily during the first few epochs, reaching its peak around epoch 3. After that, it starts to fluctuate, with a noticeable drop after epoch 3, but shows some recovery by epoch 9.
   Analysis:
   - **Initial Improvement:** In the beginning, the network is clearly learning useful patterns from the data, which is why we see accuracy steadily increasing.
   - **Fluctuations:** The drop in accuracy after epoch 3 could be due to overfitting or issues with the optimization process. This could happen if the dataset is small, leading the model to memorize rather than generalize. It might also just be natural randomness during training.
   - **Recovery:** Toward the end of training, the accuracy picks up again, which suggests that the model is finding a way to generalize better as it continues to train.
2. **Loss Plot (Bottom Panel)**
   Observation: The loss decreases consistently throughout the training, although there are a few minor fluctuations.
   Analysis:
   - **Decreasing Loss:** This is a positive sign, showing that the network is optimizing effectively and learning the underlying patterns.
   - **Minor Fluctuations:** These could be caused by noise in the data, random changes in mini-batches, or the model getting stuck briefly in local minima during the optimization process.
3. **Key Metrics from the Results Panel**
   - **Training Accuracy:** Although the exact final training accuracy isn't explicitly shown, it looks to be close to 90% based on the plot by the end of training.
   - **Validation Accuracy:** There's no validation data used in this training process, which is a bit of a limitation. Validation accuracy is really important for understanding how well the model will perform on new, unseen data. Without it, it's difficult to tell if the model is overfitting.
   - **Elapsed Time:** Training took 8 seconds, which suggests that the model is lightweight, with relatively low computational demands. It also hints at a small dataset or limited hardware resources.
   - **Training Configuration:** The model was trained for 10 epochs, with a constant learning rate of 0.001. This learning rate seems to be working well since the loss is decreasing steadily without any signs of divergence.
4. **Potential Issues**
   - **No Validation Data:** Without validation data, it's impossible to judge how well the model will generalize to new data. The fluctuations in accuracy could be a sign of overfitting.

- **Accuracy Drop:** The accuracy drop after epoch 3 might indicate overfitting or instability in the learning process. To address this, adding regularization techniques like dropout or L2 regularization could help, as would having more training data.
- **Dataset Size:** The fluctuations in the accuracy plot and the quick training time suggest that the dataset might be quite small. Training with a larger dataset would likely produce smoother trends and better results.

Summary

Overall, the model seems to perform well during training, reaching an accuracy of about 90%. However, the lack of validation data and the fluctuations in accuracy are causes for concern, pointing to potential overfitting or instability. Including validation data, using regularization methods, and expanding the dataset would likely lead to better generalization and a more robust model.

The following provides a detailed explanation of two MATLAB scripts, **Extract.m** and **SVM.m**. These scripts are used for processing audio files, extracting important audio features, and training a machine learning model to classify these features. Below, each script is explained step by step to clarify how they work and their potential applications.

Extract.m

The MATLAB script **Extract.m** is designed to process audio files from a directory, extract various audio features, and save them in .mat files for future use. Let me walk you through each step in more detail:

1. Setup and Initialization

- The script begins by defining the input and output folders. The **inputFolder(**`"PC-GITA_per_task_44100Hz\PC-GITA_per_task_44100Hz\sentences\laura\sin normalizar\HC"`**)** is the directory containing the .wav audio files that need to be processed, while the **outputFolder** is where the extracted features will be saved. (`"Extracted features"`)
- If the **outputFolder** doesn't exist, the script creates it using `mkdir`, ensuring that the output can be properly saved.
- The script then uses the `dir` function to retrieve a list of all .wav files in the input folder, ensuring that only the intended audio files are processed.

2. Audio File Loop

- The script processes each .wav file found in the **audioFiles** array. For each file, it uses the `audioread` function to load the audio data.

- **y** represents the audio signal itself.
- **fs** is the sampling frequency, which is essential for accurate feature extraction.

3. Feature Extraction

- The script extracts three types of features: MFCC, Delta, and Delta-Delta.

3.1. MFCC Extraction

- **MFCC (Mel-Frequency Cepstral Coefficients)** capture the spectral envelope of the audio signal and are widely used in speech and audio processing.
  - The script computes **13 MFCC coefficients** (set by `numCoeffs` = 13).
  - It defines a **window length** of 25 ms (using `0.025 * fs`) and an **overlap length** of 15 ms (`0.015 * fs`), creating a Hamming window to smooth the signal before applying the FFT.
  - The `mfcc` function then computes MFCC features for each frame of the audio.

3.2. Delta and Delta-Delta Features

- **Delta Features** represent the first derivative of the MFCC coefficients across time, highlighting changes between frames.
  - These are computed using a finite difference filter (`deltaFilter = [1 0 -1] / 2`).
- **Delta-Delta Features** represent the second derivative, effectively capturing the rate of change of the delta features. This is done by applying the same filter to the delta values.

3.3. Combining Features

- The MFCC, Delta, and Delta-Delta features are concatenated into a **combined feature matrix**. Each row of the matrix represents a frame of the audio, while the columns represent MFCC, Delta, and Delta-Delta coefficients.

4. Save Features

- The output file is named based on the original audio file, with `_features.mat` appended to it (e.g., `audio1_features.mat`).
- The script then uses the `save` function to store the **featureVector** into a .mat file in the output folder.

5. Progress Display

- After processing each file, the script outputs a message indicating that the feature vector has been saved successfully (e.g., "Saved feature vector for file: audio1.wav to D:...").

6. Final Message

- Once all files have been processed, the script displays a final message indicating that feature extraction is complete for all files.

Key Features and Benefits

- **Comprehensive Feature Extraction**: MFCCs capture spectral characteristics, while Delta and Delta-Delta features add temporal information.
- **Frame-Based Processing**: By processing the audio in frames, the script captures short-term features effectively.
- **Automated Workflow**: All audio files are processed automatically, and the features are saved in a structured format suitable for machine learning.
- **Scalability**: The script is capable of handling multiple files in a batch, making it suitable for large datasets.

Potential Applications

- **Speech Recognition**: The extracted features can be used as input for speech recognition models.
- **Emotion Recognition**: Temporal features help detect changes in pitch and tone, which are useful for emotion detection.
- **Speaker Identification**: MFCCs are commonly used to distinguish between different speakers.
- **Audio Classification**: Suitable for tasks like music genre classification or environmental sound recognition.

SVM.m

The MATLAB script **SVM.m** is used to load the extracted feature vectors from .mat files, assign labels, train an SVM model, and evaluate its performance. Here is a step-by-step explanation:

1. Feature Loading

- The script assumes that the **featureVector** variables are stored in .mat files located in the specified **featureDir**.
- It loads each **featureVector** and appends them into a combined matrix (**featureVectors**).

2. Label Assignment

- Labels are assigned dynamically based on the file name.
    - If the filename contains 'pd', the label is set to 1; otherwise, it is set to 0.
    - Labels are replicated to match the number of frames in each file, ensuring that each frame has a corresponding label.
- The script also checks that the number of labels matches the number of rows in **featureVectors** to ensure consistency.

3. Data Partitioning

- The data is split into training and testing sets using a hold-out method, reserving **20%** of the data for testing.
- The indices for training and testing data are retrieved using `cvpartition`, and corresponding data subsets are created.

4. SVM Training

- A **linear SVM** is trained using `fitcsvm` with standardized input features (mean = 0, variance = 1).
- **SVM** is chosen because of its robustness with high-dimensional feature vectors and moderate-sized datasets.

5. Model Evaluation

- The trained model is used to predict the labels for the test set.
- **Accuracy** is calculated as the percentage of correctly predicted labels over the total test labels, providing a basic measure of the model's performance.

6. Output

- The script displays the test accuracy, for example, "Test Accuracy: 85.00%", to indicate how well the model performed on unseen data.

Analysis and Observations

- **Feature Concatenation and Labeling**: The script's approach to combining all features into a single matrix allows for easy scaling but relies on consistent file naming conventions.
- **Data Partitioning**: The hold-out method works well for quick testing, but cross-validation could provide a more robust evaluation.
- **SVM Performance**: The use of a linear kernel suggests that the data is assumed to be linearly separable. If not, using a different kernel might improve results. Standardizing features also ensures they are on a similar scale, which improves SVM efficiency.
- **Potential Issues**: If the dataset has class imbalance, accuracy might not provide a full picture. Precision, recall, and F1-score are suggested for a better evaluation of the model's performance.

In summary, these scripts provide a robust pipeline for processing audio data, extracting meaningful features, and training a machine learning model for classification. The **Extract.m** script focuses on feature extraction, while **SVM.m** handles model training and evaluation. Together, they can be used for a range of applications like speech recognition, emotion analysis, and audio classification, offering an efficient way to process and analyze audio data.