POLITECNICO DI TORINO

Master's Degree in
Data Science And Engineering

Master Thesis

Explainable AI for Speech Data. From words to phoneme.

Supervisor
prof. Eliana Pastor
prof. Alkis Koudounas

Candidate
Raul Gatto

Academic Year 2024-2025

Summary

In recent year, automatic speech recognition (ASR) systems have seen drastic improvement in accuracy, driven by deep learning advancements. Nonetheless, their black-box nature is reduces trust and debug possibility, making it hard for users and developers to gain insights into how raw audio signals are transformed into text-like outputs and how these outputs are obtained. This opacity could also hide errors or biased outputs, making their deployment unfair and unreliable. In this context, **Explainable AI (XAI)** tries to cover this gap by producing human-understandable explanations of the model behavior.

Current ASR explanations have stopped at word-level explanations, hiding what might be found only at phonetic level. A word-level explanation attributes importance to each word, indicating **how much the results prediction shifts** when omitting or perturbing a word. This granularity can give good insights, but to gain the full picture we can explore further and explore phonemes and morphemes (groups of phonemes). This can be particularly useful for situations where a singular phonetic change can alter the meaning or intent of a sentence.

This thesis aims to **transition from the word-level down to phoneme-level**. To achieve it, we introduce a framework for generating explanations down to the phoneme level with high precision. We adapt three popular post-hoc, model-agnostic explainers (Leave-One-Out, LIME and SHAP) as to mask individual phoneme segments rather than entire words. Phonemes can also be perturbed to modify the prosodic features (e.g. pitch, duration, noise) to study how paralinguistic features influence predictions. Another implementation is the sliding-window aggregation, which can group adjacent phonemes into syllables or morphemes. All these configurations can be adjusted to explore explanations at different granularities, and by comparing results we can see where certain patterns start to form. The process saves the original classification output for a certain audio, then converts it into a time-aligned phoneme sequence. The explainers then mask the features, creating multiple versions of the original audio with a feature removed. Each new audio created is passed through the classifier and the results are compared to the original output, obtaining the individual **importance score** for each feature.

Three different datasets have been used to test the functionality of this program. Fluent Speech Commands (FSC) is composed of English sentences for smart-home appliances with three different intent classification. Then, ITALIC is used to test the framework with a different phoneme alignment library for Italian intent classification audios. Similarly, Speech MASSIVE is used to test the model for other languages, such as German and French.

Through qualitative and quantitative metrics, this thesis demonstrates that phoneme-level explanations can confirm some expectations, while showing at a finer granularity which features the system is using to arrive to its conclusions, confirming that it does not necessarily use the same human patterns and clues to arrive to a certain output.

Contents

List of Tables 7

List of Figures 9

1 Introduction 15
- 1.1 Research Motivation . . . 15
- 1.2 Problem Statement . . . 15
- 1.3 Objectives . . . 16
- 1.4 Structure . . . 17

2 Background 19
- 2.1 What is Speech? . . . 19
  - 2.1.1 Why Study Speech Processing? . . . 19
  - 2.1.2 From Speech Processing to Recognition Models . . . 19
- 2.2 Speech Recognition Models . . . 20
  - 2.2.1 Hidden Markov Models (HMMs) and Guassian Mixture Models (GMMs) . . . 20
- 2.3 Deep Learning . . . 21
  - 2.3.1 Convolutional Neural Networks (CNNs) . . . 21
  - 2.3.2 Recurrent Neural Networks (RNNs) . . . 22
  - 2.3.3 Transformers . . . 22
- 2.4 Current Limitation of ASR . . . 22
  - 2.4.1 Biases and Robustness . . . 22
  - 2.4.2 Hallucinations . . . 24
  - 2.4.3 Black-Box Models . . . 24

3 Related Work 25
- 3.1 Explainable AI in Automatic Speech Recognition . . . 25
  - 3.1.1 Adaptation of XAI Techniques to ASR . . . 26
  - 3.1.2 Visualization and Interpretation Strategies . . . 26
  - 3.1.3 Word-level analysis . . . 27
  - 3.1.4 Filling The Gaps . . . 27

4 Explanation Methods in Artificial Intelligence 29
4.1 Introduction to Explainable Artificial Intelligence . . . 29
4.1.1 By-Design vs. Post-Hoc Explainability . . . 30
4.1.2 Accuracy-explainability Trade-off . . . 31
4.1.3 Faithfulness and Plausibility . . . 32
4.1.4 Scope of explainability . . . 33
4.1.5 Post-hoc explainability: model-dependent vs model-agnostic solutions 33
4.2 Leave-One-Out (LOO) or Leave-One-feature-Out (LOFO) . . . 37
4.3 Local Interpretable Model-Agnostic Explanations (LIME) . . . 38
4.4 SHapley Additive exPlanations (SHAP) . . . 40
4.4.1 Shapley Values . . . 40
4.4.2 SHAP . . . 42
4.5 Paralinguistic Attributions . . . 49
4.6 Implementation of Interpretable Techniques . . . 50
4.6.1 Leave-One-Out (LOO) . . . 51
4.6.2 Local Interpretable Model-Agnostic Explanations (LIME) . . . 53
4.6.3 SHapley Additive ExPlanations (SHAP) . . . 54
4.6.4 Transcription and Aggregation . . . 55
4.6.5 Evaluation Metrics . . . 58

5 Experimental Results 59
5.1 Datasets . . . 59
5.1.1 Fluent Speech Commands (FSC) . . . 59
5.1.2 ITALian Intent Classification (ITALIC) and Speech MASSIVE . . . 60
5.1.3 Interactive Emotional Dyadic Motion Capture (IEMOCAP) . . . 61

6 Results 63
6.1 FSC Sample Analysis . . . 63
6.2 Multi-language analysis . . . 71

7 Conclusion 77

List of Tables

6.1 **Word-level explanations with LOO** for the FSC dataset sample, a higher value and deeper color corresponds to more relevance for the prediction. 64

6.2 **Phoneme-level explanations with LOO** for the FSC dataset sample, a higher value and deeper color corresponds to more relevance for the prediction. To fit the page, the table was visually split in two rows, and the vertical lines help distinguish between words. Results are approximated to 3 decimals. 65

6.3 **Phoneme-level explanations with LIME** for the FSC dataset sample, a higher value and deeper color corresponds to more relevance for the prediction. To fit the page, the table was visually split in two rows, and the vertical lines help distinguish between words. Results are approximated to 3 decimals. 68

6.4 **Phoneme-level explanations with SHAP** for the FSC dataset sample, a higher value and deeper color corresponds to more relevance for the prediction. To fit the page, the table was visually split in two rows, and the vertical lines help distinguish between words. Results are approximated to 3 decimals. 68

6.5 Evaluation for the LOO explanation for the FSC sample, where a deeper green indicates a good result and orange a result that could be improved. . 69

6.6 Evaluation for the LOO explanation for the FSC sample, where a deeper green indicates a good result and orange a result that could be improved. . 70

6.7 Evaluation for the SHAP explanation for the FSC sample, where a deeper green indicates a good result and orange a result that could be improved. . 70

6.8 Phoneme-level explanations with LOO for the FSC dataset sample, where higher values and deeper colors indicate more relevance. The table is split in two to fit the page. Parameters: `windows_size = 2, respect_word_boundaries = True, sliding = False`. 71

6.9 Phoneme-level explanations with SHAP for the FSC dataset sample, where higher values and deeper colors indicate more relevance. The table is split in two for readability. Parameters: `windows_size = 2, respect_word_boundaries = True, sliding = False`. 71

6.10 Phoneme-level explanations with LOO for the FSC dataset sample, where higher values and deeper colors indicate more relevance. The table is split in two for readability. Parameters: `windows_size = 2, respect_word_boundaries = True, sliding = True`. 72

List of Figures

2.1 Basic pipeline for speech recognition. 20
2.2 Deep neural network representation 21
2.3 Disparities in a speaker identification task, showing better performance for women and different performance based on race. 23
4.1 Diagram showing the stakeholders and the reasons they might need explainable systems, from [1]. 30
4.2 Transparent decision tree used to categorize the iris flower based on its features. 31
4.3 Deep trees lose their interpretability. 31
4.4 The trade-off between accuracy and explainability 32
4.5 Grad-CAM identifying different animals. 34
4.6 Main steps to achieve an interpretable explanation 35
4.7 Global insights of a multi-class model, showing feature importance for each possible class output. 47
4.8 Global insights obtained from a beeswarm plot. 48
4.9 Feature dependence for the feature *Age* and the feature *Education_Num*. . 48
4.10 Graph showing the main steps of how the program achieves explanations. . 51
6.1 Conversion between ARPAbet vowels and IPA vowels. 64
6.2 The figure shows the timestamp of the sentence, each word and phoneme. . 66
6.3 Waveform of the FSC audio sample with phoneme alignment and importance scores color-coded. 67
6.4 Phoneme-level explanations with LOO for the ITALICS dataset sample *"Che tempo fa questa settimana?"*, where higher values indicate more relevance. 72
6.5 Phoneme-level explanations with LOO for the ITALICS dataset sample *"Mi piacciono le canzoni di Bocelli"*, where higher values indicate more relevance. Parameters: `windows_size = 2, respect_word_boundaries = True, sliding = False`. 73
6.6 Phoneme-level explanations with LOO for the Speech MASSIVE dataset sample *"Wie wird diese Woche das Wetter?"*, where higher values indicate more relevance. 74

6.7 Phoneme-level explanations with LOO for the Speech MASSIVE dataset sample *"Ich mag Lindenberg-Lieder"*, where higher values indicate more relevance. Parameters: `windows_size = 2, respect_word_boundaries = True, sliding = False`. 74
6.8 Phoneme-level explanations with LOO for the ITALICS dataset sample *"Quelle est la météo pour cette semaine?"*, where higher values indicate more relevance. 75
6.9 Phoneme-level explanations with LOO for the Speech MASSIVE dataset sample *"J'aime les chansons de Jacques Brel"*, where higher values indicate more relevance. Parameters: `windows_size = 2, respect_word_boundaries = True, sliding = False`. 75

Acronyms

ASR Automatic Speech Recognition.

CNNs Convolutional Neural Networks.

FSC Fluenti Speech Commands.

GDPR General Data Protection Regulation.

GMMs Gaussian Mixture Models.

HMMs Hidden Markov Models.

ICE Individual Conditional Expectation.

IEMOCAP Interactive Emotional Dyadic Motion Capture.

ITALIC ITALian Intent Classification.

LIME Local Interpretable Model-Agnostic Explanations.

LOFO Leave-One-feature-Out.

LOO Leave-One-Out.

NLP Neural Language Processing.

PDPs Partial Dependence Plot.

RNNs Recurrent Neural Networks.

SHAP SHapley Additive exPlanations.

SVM Support Vector Machine.

XAI Explainable AI.

Chapter 1

Introduction

1.1 Research Motivation

Speech recognition systems are everywhere in our daily lives, and as these systems become more and more sophisticated and deployed in critical applications, understanding how they make decisions has become increasingly important. Traditionally, speech recognition models can be highly effective but they also operate as "black boxes" models, making decisions that are difficult for humans to understand, interpret or verify. This is especially true for the application of deep learning and neural networks, which brought great advancements in the field at the cost of complexity and opacity of these models. This becomes even more important when they make errors or introduce biases in their decisions [2].

Interpreting these models is hard because speech is complex, containing multiple layers of information such as raw acoustic features, semantic meaning and sounds such as words and phonemes. Current approaches typically work at the word level, providing a good enough insight into which words influence the model predictions the most. However, words can have insufficient granularity if we want to really understand and deep dive into the model behavior and which information are really important, as some of it could be hidden at higher levels of granularity.

For example, consider a speech emotion recognition system that can identify anger in a speaker. A word-level explanation might be able to tell us which words contributed the most to this classification, but it cannot tell us if the model is responding to some specific phonetic features, such as pitch or speed or even accents.

1.2 Problem Statement

The fundamental problem this thesis tries to address is the gap between word-level and phoneme-level interpretability in speech recognition systems. Current explanations methods can provide lots of insights for how vision and tabular data operate, but speech

model explanations still suffer from some limitations such as:

1. **Granularity**: As introduced, existing methods typically treat words as atomic units, which causes explanations to miss sub-words patterns that can influence model decisions. This is especially important in languages where phonetic features can assign totally different meanings to certain words.

2. **Perturbation**: Perturbation-based methods are usually applied to the whole audio, making it difficult to understand both which word carries the most information and which type of perturbation can affect the classification the most.

3. **Segmentation**: Moving from audio or word level to smaller units such as phoneme is a challenge by itself, because on one hand we might lack precision explanations at word level but on the other hand we might introduce errors in the segmentation at phoneme level, skewing explanations.

4. **Lack of datasets** with proper timestamps for phonemes.

1.3 Objectives

The aim of this thesis is to address the limitations and enhance the interpretability of speech recognition models through fine-grained analysis at the phoneme level, specifically by:

1. **Improving the granularity**, to move from word-level to phoneme-level segmentation.

2. **Adapting the explainability methods** to work at phoneme-level since the existing explanation frameworks (LOO, LIME and Gradient-based) work at the word level.

3. **Adapting the perturbation techniques** that work at the audio level and increase their precision by perturbing words or phonemes.

4. Introduce a **phoneme aggregation** method while respecting word boundaries to gain more insights on how they interact with each other.

5. **Compare and visualize** the results of these explanations across different levels of granularity.

The goal is to give a deeper understanding on how these models work and enable more reliable and transparent speech recognition systems, while also maintaining the ability to quickly switch between the different granularities. The study of the benefits of phoneme-level explanations will demonstrate their utility in certain scenarios, while also discussing some of the limitations of this approach.

1.4 Structure

This thesis will be organized in the following chapters:

1. **Chapter 1. Introduction** - Presents the motivation, the problem statement, the objective and the contribution of the research.
2. **Chapter 2. Background** - Reviews existing work in speech model interpretability, discusses limitations and examines the relevant technical foundations.
3. **Chapter 3. Explainability**
4. **Chapter 4. Methodology and Implementation** – Details the theoretical framework for the explanation methods, the approach to perturbations and the integration of multiple levels of granularity.
5. **Chapter 5. Experiments and Results** - Presents the datasets, the settings and an explanation for the choices made for certain parameters. Also include the results and the analysis of the effects of the different explanations, perturbations and levels.
6. **Chapter 6. Conclusion** - Summarizes key findings and discuss the outlines for future research directions.

Chapter 2

Background

This first chapter will introduce the basics of speech processing and the reasons why it is important to study it. We will then outline the evolution of speech recognition models and their current limitations.

2.1 What is Speech?

Speech is a complex form of human communication that involves the production of vocal sounds. These sounds are structured and organized to enable the exchange of information, emotions, and intentions between individuals. Unique to humans, speech is a fundamental aspect of society that facilitates interpersonal communication and the sharing of ideas, requiring a shared understanding of a language that allows individuals to share concepts efficiently. At the base of human-computer interaction through the means of speech, we find the topic of speech processing.

2.1.1 Why Study Speech Processing?

Speech processing is a critical area of research and application within the fields of artificial intelligence (AI) and computer science. It involves the analysis, interpretation, and generation of human speech through a variety of models. Speech processing has gained importance due to its wide range of applications, from voice recognition to artificial speech synthesis because, unlike writing or typing, spoken language allows for seamless communication. By enabling machines to comprehend and respond to spoken language, speech processing improves the development of intelligent systems capable of human-like interactions.

2.1.2 From Speech Processing to Recognition Models

Speech processing includes multiple stages, from capturing the acoustic signal to the interpretation of their meaning. A simplified pipeline can be seen in (Figure 2.1).

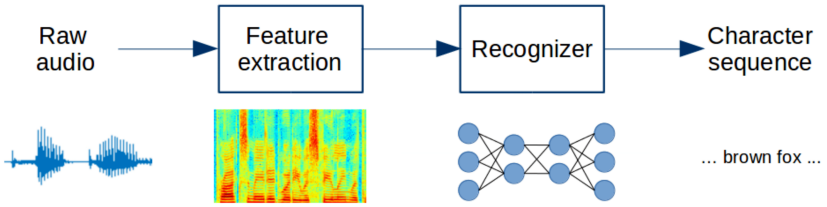

Figure 2.1. Basic pipeline for speech recognition.

At its core is speech recognition, the ability to convert spoken audio into written text through an internal representation of the features of the audio. Each model can work in a different way, and understanding them can help shifting their limitations forward.

2.2 Speech Recognition Models

In this section, we will provide an overview on speech recognition models, and discuss the challenges in their interpretability.

Speech recognition models are advanced systems designed to convert spoken language into text, enabling human-computer interaction through the means of voice. This technology has improved a lot from early techniques that used pattern matching techniques [3], that tried to measure the distance between words and a predefined stored template, to the most recent **deep learning** techniques that work with multiple hidden layers capable of modeling complex non-linear relationships, translating to better feature handling.

Speech recognition is used in a variety of application and ranges from automated controls and interaction with virtual assistants to accessibility tools for individuals with disabilities, and more. Despite the advancements, there are also some challenges due to the differences between individuals and the way everyone speaks: different accents, pitches, speech impairments, background noises, code-switching... Moreover, the dynamic nature of language with an evolving vocabulary necessitates continuous advancements in Automatic Speech Recognition (ASR) systems.

2.2.1 Hidden Markov Models (HMMs) and Guassian Mixture Models (GMMs)

Before the application of deep learning, *Hidden Markov Models* (HMMs) combined with *Gaussian Mixture Models* (GMMs) **were the backbone of ASR**. HMMs captured the **sequential structure of speech** using probabilistic state transitions with the likelihood of words and sentences sequences, combined with the GMMs which provided statistical models for the acoustic features. In other words, HMMs analyzed speech as a sequence of sounds predicting the most likely words and phrases based on probabilities, while GMMs helped distinguish different sounds by modeling speech patterns. Together, they allowed

ASR systems to predict phonemes, syllables, words, and sentences.

Even though they achieved good results, they **struggled with long sentences** and complex speeches due to their probabilistic nature. HMM were gradually integrated into the first deep learning models, and while most models today have moved beyond pure HMM architectures, their principles influenced the modern approach to ASR.

2.3 Deep Learning

The development of deep learning was a big leap forward for speech recognition, as it did not require the need for a phoneme dictionary and improved robustness against noise, speaker characteristic, and acoustic conditions. Unlike earlier methods, deep learning models can learn directly from raw data, making them highly versatile.

Deep learning models are structured using **multiple layers of artificial neurons**, where each layer extracts more complex features from the input. The first levels capture low-level details, such as frequency components, while deeper layers recognize features like phonemes, syllables and words recognizable by humans. By utilizing large datasets and advanced algorithms, they can **learn complex patterns in speech** without the need of feature manipulation.

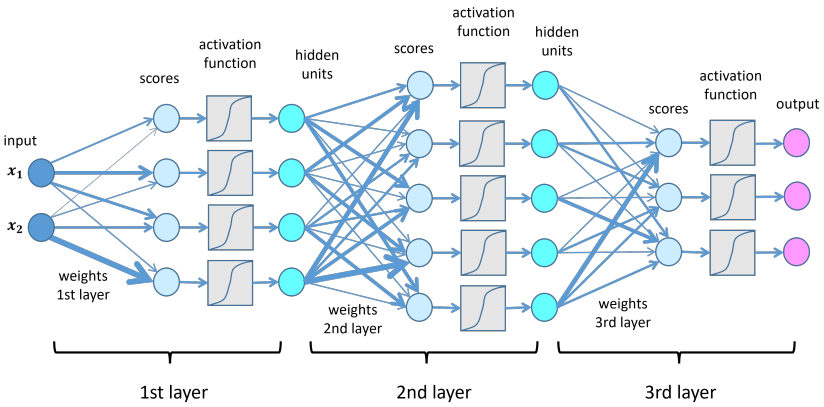

Figure 2.2. Deep neural network representation

The primary architectures involved in ASR are *Convolutional Neural Networks (CNNs)*, *Recurrent Neural Networks (RNNs)*, and *Transformers*.

2.3.1 Convolutional Neural Networks (CNNs)

CNNs, which are generally used for **image processing**, have also been applied to speech recognition [4]. This is due to the ability to visualize audio signals as **spectrograms**, where the x-axis represent the time and the y-axis the frequency, and the color represents amplitude. They are good at analyzing spectrograms because they can detect

patterns such as pitch variations, phoneme transitions and other frequency characteristics. This allows CNNs to extract high-level acoustic features, such as formants and harmonics, and pass this information to other models.

CNNs are generally not used alone for ASR anymore, but when combined with RNNs or transformers, CNNs can enhance feature extraction and improve ASR performance, as seen in Conformers [5] from Google.

2.3.2 Recurrent Neural Networks (RNNs)

RNNs, and particularly the *Long Short-Term Memory* (LSTM) variants, are good at processing sequential data by **maintaining memory of previous inputs** and capture the patterns of speech data, which develops over time.

RNNs by themselves suffer from the problem of **vanishing gradient**, resulting in RNNs "forgetting" parts of the input that appeared earlier when processing longer sequences. LSTM were introduced to allow RNNs to retain information over longer time spans, improving the architecture and the accuracy.

Even though RNNs and LSTM have significantly improved ASR, they have been replaced by Transformer-based models.

2.3.3 Transformers

Transformers were introduced in 2017 and revolutionized deep learning methods thanks to the **attention mechanism** [6]. Unlike RNNs, which process data sequentially, Transformers can parallelize computations and use an attention mechanism to analyze the entire input at once.

The attention allows the models to focus on the most relevant part of the input dynamically, consider the entire sentence context without suffering from vanishing gradients and capturing short-term and long-term dependencies in the input sequence.

Many of the the state-of-the-art ASR models are based on Transformers, such as *Wav2Vec 2.0* from Meta [7], that learns representation from raw audio, *Conformers* from Google [5] and *Whisper* from OpenAI [8].

2.4 Current Limitation of ASR

While Automatic Speech Recognition systems have seen good improvements in recent years, they still face several significant challenges that impact their accuracy, reliability and real-world applications.

2.4.1 Biases and Robustness

One of the most widely known challenges of AI systems is the presence of **data biases**, including ASR systems. AIs need to be trained on large amounts of data, and since this

data comes from the real-world, they often reflect societal imbalances. Biases can manifest in multiple ways, such as disparities in performance across different genders, ages, ethnicity or accents. Numerous studies have explored methods to identify, quantify and mitigate these biases [9], [10], [11], [12], [13].

The issue of bias is strictly linked to the **robustness** of the model, which refers to the ability of accurately transcribing speech under various conditions, speakers, and accents. This connection comes from the natural variation of human speech, where factors like pronunciation, accent, and paralinguistic features (such as pitch, speed, rhythm) can impact the model performances. For example, a model trained primarily on native American English speakers, might struggle when dealing with strong accents or non-native speakers, since the model is not familiar with their pronunciation and could create wrong transcriptions (Figure 2.3, from [14]).

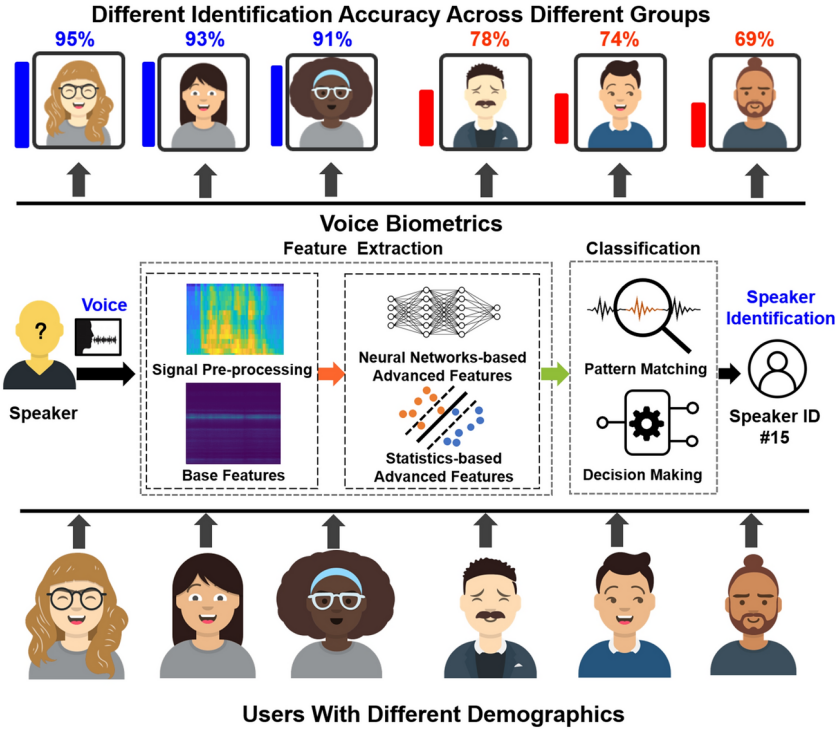

Figure 2.3. Disparities in a speaker identification task, showing better performance for women and different performance based on race.

A similar issue is represented by **out-of-vocabulary words**, words that the system have never seen during training, including slang, proper names or technical vocabulary. For example, a ASR model trained for automated captioning in a general context would not be fit for the deployment in a social media, where users communicate with informal language, slang and code-switching between languages.

To address these challenges and build systems that are more robust and fair, researchers need to develop **more diverse and representative training datasets** from a wider variety of speakers. On top of this, it's important to consider where the model will be deployed and who will be interacting with it. This will ensure better performance in real-world application and for a more diverse crowd.

2.4.2 Hallucinations

Hallucinations are a type of error produced by deep neural networks, widely studied in natural language processing. Recently, hallucinations have also been studied for ASR systems, defined as transcriptions that are linguistically coherent and fluid, but that have little to no relation to the source sound. Wrong outputs impact the credibility of the system, since traditional methods cannot differentiate between hallucinations and accurate transcriptions.

Hallucinations often arise due to low-quality training data, where incorrect annotations or noise can introduce wrong correlations in the learned model. This issue underscores the importance of high-quality datasets and proper preprocessing. The susceptibility of these models to create hallucinations has been studied through perturbation-based techniques [15].

2.4.3 Black-Box Models

Even though neural network can have as low as two layers, ASR systems generally employ deep architectures and have a very large number of parameters, ranging from the thousands to the billions. This is at the base of the problems regarding the interpretability of these **"black-box" models**. Unlike rule-based systems, that have defined and explicit decision steps, deep learning models operate with multiple layers of abstraction and non-linear transformations, meaning that they have an **opaque decision making** process.

The general pipeline of an ASR model starts with feature extraction, transforming the output into an internal representation that is comprehensible to the model. These features are then processed through deep neural layers, where weights and transformations influence the final output. This internal representation is not directly interpretable, increasing the gap between human-understandable features and the model abstract feature space.

Improving interpretability and being able to gain insights of the inner workings of these black-box models is crucial for ASR, leading also to improvements in biases identification, user trust and even helping in debugging errors. Some possible solutions are discussed in the following chapters.

Chapter 3

Related Work

In the recent years, Artificial Intelligence (AI) has gained a lot of interest, especially thanks to the popularization of Large Language Models (LLMs) like ChatGPT. AI, if used correctly, can deliver an improvement in the outcome of many different sectors. For this to happen, the machine learning community needs to address the barrier of explainability. This stems from the inherent problem of AIs: using sub-symbolism, such as ensembles or neural networks which humans cannot easily understand like rule-based systems. As we have already introduced, eXplainable AI (XAI) is the field that tries and explain how these new AI systems work, and it is acknowledged as one of the most important features for safe deployment of AI models [1].

In this chapter we will briefly present the existing literature regarding XAI, particularly in the field of Automatic Speech Recognition.

3.1 Explainable AI in Automatic Speech Recognition

The advancements of AI even in the field of ASR ([16]) has made it clear that even speech requires deep researches to clarify the inner workings of these increasingly sophisticated models. As previously seen, initially ASR systems relied on Hidden Markov Models (HMMs) combined with Gaussian Mixture Models (GMMs), which were then substituted because of their inability to capture and manage complex and long sequence of speech data. With the advent of deep learning, ASR performance significantly increased but they also gained complexity, acting now effectively as black-boxes, thus losing transparency in their decision making.

Some of the immediate challenges that researchers have to address when implementing the already existing methods for ASR is the nature of audio data: both temporal and sequential, unlike static images or text data. This means that a sentence can have temporal dependencies and variations, which makes the traditional perturbation methods less direct to implement: audio frames usually do not have inherent meaning unless considered in context. Perturbations such as zeroing or adding noise to some frames can create unrealistic audio sequences, which has to be taken into account when applying

these methods. On top of this, speech resides in a high-dimensional feature space, and methods like SHAP that require lots of calculations to assess all possible combinations of features. This makes the method even harder to implement, especially as granularity increases, making optimized methods such as Kernel SHAP, Deep SHAP and other even more important to research and develop [17].

3.1.1 Adaptation of XAI Techniques to ASR

To address the transparency issue, researchers have adapted the already existing post-hoc explainability methods that were originally designed for images and texts. Some of these are Local Interpretable Model-Agnostic Explanations (LIME) [18] and SHapley Additive exPlanations (SHAP) [17]. Particularly, LIME can be adapted in a way that enables to identify the minimal and sufficient subset of audio features that can identify the correct output, as seen in [19]. This allows the user to get frame-level insights, to precisely identify the most important features. Similarly, also SHAP has been extended to work within ASR systems, which also allows the visualization similarly to the already existing implementations [**?**].

LIME was originally developed for images and works by generating perturbations of the input and evaluating how these changes affect the output of the model. In the case of ASR, the input is made of audio frames, which are the raw unit of data of speech at specific time intervals, so the method had to be adapted to be able to move from the pixel-feature space to this new feature-space. The method also used to mask pixels, and now needs a new way to mask audio frames. This creates mutated versions of the original audio input that are then fed into the speech classification model and the output is then analyzed once again, to evaluate if the changes in those audio frames affected the output [19]. Similarly, they also adapted the Causal method [20], which was also originally proposed for image classifiers. This method focuses on understanding causal relationships between audio frames and the output, but works with the concept of "superframes". This means that instead of perturbating single frames, it uses groups of frames and evaluates how changes in multiple frames affect the model's decisions.

As seen in [21], multiple studies use Layer-wise Relevance Propagation (LRP) [22], which was also proposed for image classification and then adapted to explain audio tasks. This method works by backtracking the relevance of of the prediction through the layers of the network, and it uses spectrograms and waveforms of the audio data to identify which parts of the spectrum are the most relevant for the output. LRP can also be used to derive relevance scores for individual samples of an input, as for [23].

3.1.2 Visualization and Interpretation Strategies

For what it concerns the visualization of explanation for audios, some works represent explanations using time-frequency heatmaps over the spectrogram of an audio, which means highlighting areas of importance over the time and frequency representation, for example in [23], [24], [25], [21], or even heatmaps over some terms [23]. Heatmaps and spectrogram can carry a lot of information, but they are hard to interpret for many people since they require specialized knowledge to decode them, creating a barrier for non-experts

users. Similarly, another method assigns relevance scores to audio frames, dividing the audio in data bins based on predefined time spans [26]. Some variations of LIME, like SoundLIME [27], apply this equal-width approach to segments within the time and/or frequency domains to perform music content analysis. It is important to notice that the time we choose for the creation of the segment does affect the explanations, on top of the fact that the explanations are not based on linguistic elements such as words or paralinguistic features, thus reducing the interpretability of the results.

3.1.3 Word-level analysis

In [28] we find a new explainability approach for speech classification, that analyzes speech at word-level and also addresses the role of paralinguistic features. Unlike the previous methods, their approach tries to align the audio with each transcription, and evaluates the contribution of each word segment or paralinguistic feature (e.g. pitch, noise) to the model's predictions. The work in this thesis is based on this research and adopts similar perturbation-based techniques, and methods like Leave-One-Out and LIME, implementing SHAP into our research, a well-established method in the field of XAI [29]. They also propose a user study to assess the plausibility of their explanations, also validating their new approach for visualizing explanations even to non-experts. In [30] a similar approach is used, but they test fixed-width segments of audio and align the audio with the phonemes. Unlike our work, they already had phoneme-level timestamps for the transcription, which is usually not provided and hard to obtain.

To perturb the data, different domains utilize different techniques, such as removing parts or masking them with the addition of noises, blurring, masks for vision or removing words for texts, or even average values for structured data [29]. In the case of audios, we can zero the signal to mask certain parts of it [30], add noises, modify the speed, pitch and many more.

Always in the field of XAI, multiple efforts have been made to analyze speech models at the subgroup level, trying to address and mitigate biases and fairness issues that might arise [31], [32], [33], [34], [35], [36]. This is different from the work presented in this thesis, which focuses on improving the granularity of interpretability at the individual level.

3.1.4 Filling The Gaps

Despite the advancements and the ongoing research, addressing our problem at hand had some specific challenges to overcome.

Dataset Scarcity. One of the biggest problems when dealing with phoneme-level explainability is the lack of datasets that provide accurate phoneme timestamp. While word-aligned datasets can be found, phoneme-level is hard to obtain because it requires a trained ear or the use of forced alignment tools, which add a layer of uncertainty and error possibility. One dataset, TIMIT [37], is available with phoneme-level transcription and timestamps, but it also uses a proprietary representation of phonemes instead of the International Phoneme Alphabet (IPA) or ARPABET, commonly used for machine learning

tasks, making it unusable to obtain comparison for our task. To overcome this problem, we will utilize a compatible forced aligner developed only for English and Chinese [38].

Granularity Shift. Most techniques for explainability in ASR use fixed-width audio frames or datasets with timestamps already available. Shifting from fixed-width or word-level to phoneme-level introduced a new challenge on how to obtain this very granular and precise information. Multiple forced alignment tools were not compatible with the structure of the program, such as MFA [39]. These tools can also introduce a layer of possible error, and when we extend our research outside of the English language the challenge becomes even harder. This shift also increases the complexity of the explanation space, increasing the number of features to evaluate. This can lead to an exponential increase in the time of execution of certain methods, such as SHAP, which requires the calculation of many combination of such features to obtain the importance scores.

Chapter 4

Explanation Methods in Artificial Intelligence

As Artificial Intelligence systems become increasingly complex, understanding their decision-making progress has increased in priority in the research field. Explainable AI (XAI) addresses this challenge by developing techniques to make AI systems more transparent on their inner functioning. This is as important in speech recognition systems, where the relationship between input audio and model predictions can be especially opaque.

While good progress has been made in the field of XAI for image, text and tabular data, speech presents more intricacies due to its temporal nature and the gap between continuous signals and the discrete units that we use. With some adaptation, traditional explanation models can be adapted to work for speech systems.

This section will present the main terminology and topics of XAI, the most common approaches to generating explanations for speech models, as well as presenting the challenges in phoneme adaptations for each of them.

4.1 Introduction to Explainable Artificial Intelligence

Explainability is defined as *the ability to explain the reasoning behind the decisions or predictions that the AI system makes in understandable terms to humans.* Explainability is often confused with **interpretability** but they carry subtle differences. The latter refers to the *inherent characteristic of a model to be understandable*, or "to make sense", from a human perspective, while the former is more of an acquired characteristic through external methods, with the intent of clarifying the output or its internal functions. Aside from these differences, the terms are more used interchangeably, even though to achieve full **transparency** a model should satisfy both the condition at the same time [1].

Decisions should be explained to and **understood** by those directly and indirectly affected by the AI and told in a clear and understandable way by them Figure 4.1. It should also be possible to demand a suitable explanation of the decision-making process

as for the General Data Protection Regulation (GDPR) Articles 13 and 14 [40]. All these requirements are impossible to fulfill for a black-box model such as deep networks, and the XAI field is working on improving these opaqueness to increase the trust in the models of the general consensus.

Trust is also extremely important for the implementation and the

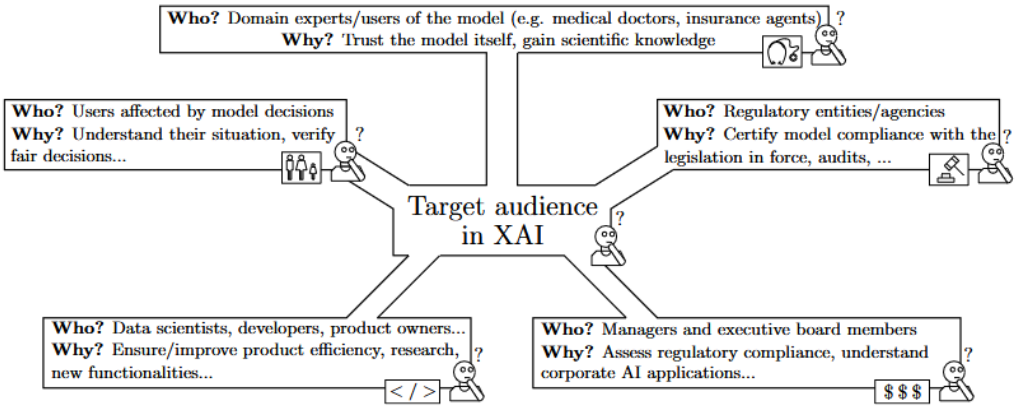

Figure 4.1. Diagram showing the stakeholders and the reasons they might need explainable systems, from [1].

4.1.1 By-Design vs. Post-Hoc Explainability

First, we can introduce the two categories of approaches when talking about explainable models.

1. **By-design explainability**: They are models that are inherently interpretable through their architecture, thus being transparent. This can include decision trees (Figure 4.2, from [41]), decision rules, regressions .

2. **Post-hoc explainability**: They generate explanations for a trained model, analyzing its behavior without modifying its internal structure. This is the focus of this thesis, since post-hoc explanations are usually applied to the high-performing black-box models and is where the majority of XAI approaches focus on.

One might think to always apply by-design explainable models, but they suffer from many drawbacks, like the accuracy-explainability trade-off. Some models require regularization (e.g. introducing some constraints so that the model remains interpretable) to target interpretability, while others, like decision trees, can get extremely deep and lose interpretability (Figure 4.3).

Another approach is the explanation-in-the-loop, which consists of training the AI with the correct prediction and an associated explanation. When a system outputs a prediction, we will also get its associated explanation [42]. This last approach introduces the need of datasets annotated with explanations, and the faithfulness to the model and

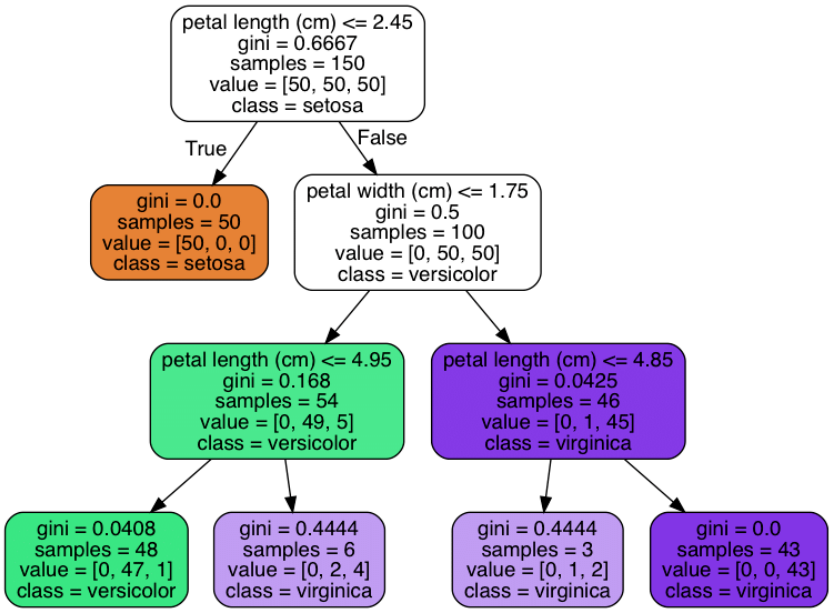

Figure 4.2. Transparent decision tree used to categorize the iris flower based on its features.

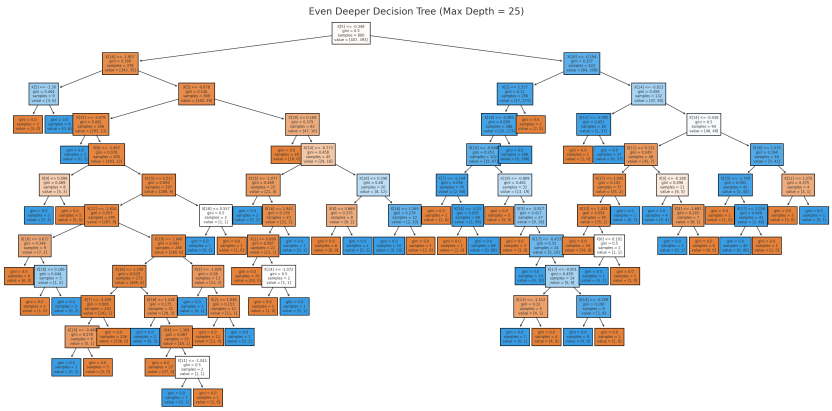

Figure 4.3. Deep trees lose their interpretability.

plausibility problems [43].

4.1.2 Accuracy-explainability Trade-off

The **accuracy-explainability trade-off** arises from the complex algorithms used in AI systems. On one hand, models that are accurate, such as deep networks, can achieve high performances on many tasks like image recognition or natural language processing

(NLP), but, as seen, they also act like "black-boxes", making it hard to understand the process to get to their result. On the other hand, simpler models like decision trees or linear regressions can offer great transparency because their decision-making paths are explicit and easy to follow, but they usually achieve less satisfying results than more complex models, as Figure 4.4 [44]. This creates the dilemma, *is it better to prioritize accuracy for better predictions or choose explainability for greater trust and transparency?*

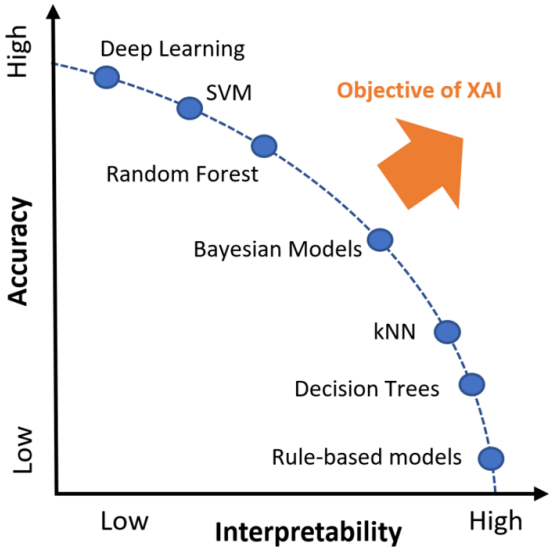

Figure 4.4. The trade-off between accuracy and explainability

A possible solution could depend on the application of the system and its context. A model deployed in healthcare should be able to provide clear reasoning for each prediction, so that the users can trust the model. This case would prefer higher explainability, so that in case of false-positives the expert can spot them with ease. Another possible solution is to use post-hoc explainability methods, which can clarify the output of a model. By carefully considering the needs and the context, developers can balance properly the need for accuracy or explainability.

4.1.3 Faithfulness and Plausibility

Other two important topics in the realm of XAI are faithfulness and plausibility.

1. **Faithfulness** indicates whether or not the explanation matches the model inner working. If an explanation is faithful, it accurately represents how a model arrived at its decision.

2. **Plausibility** indicates whether or not the explanation matches what humans expect, meaning an explanation may seem plausible to a human, even if it does not reflect the true model's inner thinking. The use of dataset annotated with explanations can help evaluate the plausibility of the model-generated ones [43].

In modern systems, faithfulness and plausibility are in an antagonistic situation, meaning that if we increase the plausibility to create user-friendly explanations, it might come at the cost of diminishing faithfulness [45]. This means that the model creates a solution that is acceptable by humans because of its logic and coherence, but it's not an accurate depiction of how the model actually arrived to that conclusion. This is due to the fact that models learn from patterns that humans might not consider, features that we do not perceive such as frequencies or noises, and mathematical operations through layers.

4.1.4 Scope of explainability

Explanations can be categorized based on their scope, ranging from insights applicable to the entire model to more granular explanations for specific instances. The main levels of explainability are:

1. **Global explanations**: Global explainability aims to provide insights into the overall behavior of the model, to explain how it works "in general". It tries to answer questions like *"Which features influence the model the most?"* by using feature importance analysis, partial dependence plots and interpretable simplified models.

2. **Subgroup explanations**: Subgroup explainability aims to provide insights into how the model behaves for a particular group (e.g. a race, an age range). This level of explanations is particularly useful when we want to test the model against disparities across different groups. A question we can try to answer is *"Does the model behave differently for a certain group?"* and if it does, try and understand the reason why with fairness metrics.

3. **Local explanations**: Local explainability aims to provide insights for individual predictions, answering questions like *"Why did the model make this particular decision?"*. Explaining a single prediction is easier than explaining an entire model, and it's also easier to understand and analyze. Many methods have been proposed, differentiating on how they represent and generate the explanation.

4.1.5 Post-hoc explainability: model-dependent vs model-agnostic solutions

When models do not meet the criteria to be classified as transparent and do not have inherent characteristics of interpretable models, external methods must be applied to it to explain its decisions. In this section, the solutions are divided in model dependent or model agnostic. These approaches differ in how they interact with the model and the level of flexibility in explaining different systems.

Model-Dependent Approaches

These methods are built for specific types of models, leveraging their internal structure to generate explanations. These models can usually provide accurate and fine-grained

explanations because they utilize details specific to that model, such as weights and gradients, but they have to be developed ad-hoc.

1. **Decision trees** naturally provide feature importance scores by analyzing how features contribute to splits.

2. **Support Vector Machines (SVMs)** were the focus for model-specific simplification techniques to make their high-dimensional decision boundaries more interpretable, but their interest declined with the rise of deep learning models [46] [1].

3. **Gradient-based** methods are used for deep learning models. Saliency maps and Grad-CAM utilize gradients to identify the most important input features for a given prediction and visualize the result on text and images (Figure 4.5) [47].

4. **Attention mechanisms** are used for transformers and attention-based neural networks to highlight the relevant parts of the input. This method also count as in-model because attention mechanisms are built-in, but there is an ongoing debate whether attention can be counted as explanation or not [48], [49], [50].

5. Through the years, multiple approaches were developed for specific networks, like TreeSHAP for tree-based models [17] (e.g. XGBoost [51]), or GNNExplainer for graph-based neural networks [52].

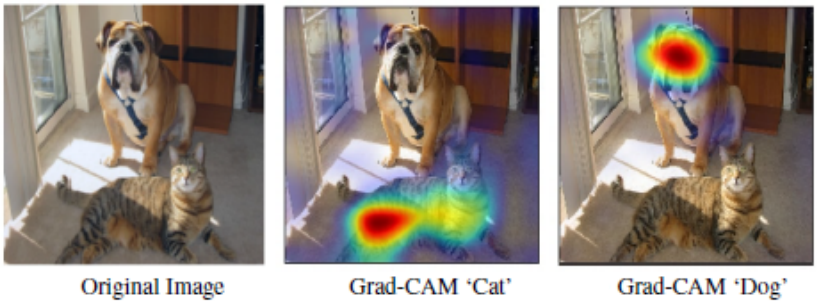

Figure 4.5. Grad-CAM identifying different animals.

Model-Agnostic Approaches

Model agnostic techniques are designed to be applicable to any model, relying on it as an oracle that output probabilities and focusing on the relationship between inputs and outputs. The intent is to extract information from its prediction procedure independently from its underlying structure [53]. Aside from the flexibility to apply each technique to a variety of models, they also offer the ability to represent the explanation in a format different than the one used in the internal structure of the model and suitable for the targets. Finally, if the case arises, it's possible to change the underlying model while

preserving the explanation method, or to compare different model for the same prediction.

Many methods have been developed, the following is a broad classification of existing methods, but many more exists and many could also be categorized under multiple classes:

1. **Simplification**: This is the broadest category of model agnostic post-hoc methods. It can use interpretable surrogate for more complex models. This is achieved by training an interpretable model with the outputs of the black-box, and trying to minimize the difference in predictions. Some of these models can suffer from over-simplification and need to be trained on outputs representative of the whole datasets. As for Figure 4.6, the black-box model is then simplified by an interpretable model, which can output a human-understandable solution.

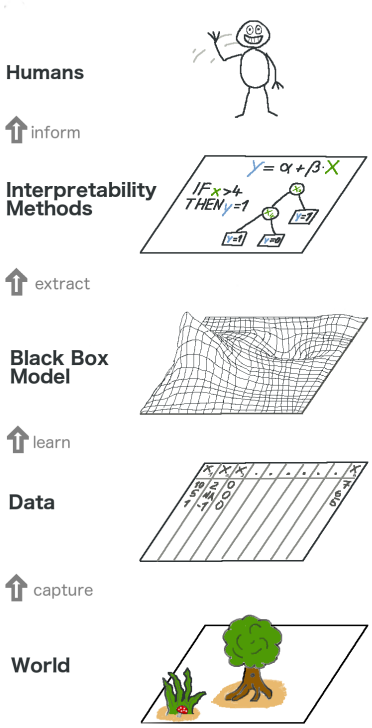

Figure 4.6. Main steps to achieve an interpretable explanation

2. **Feature Importance Methods** or **Perturbation-based Methods**: These methods aim to describe the functioning of a model by measuring the importance of each feature. They do not rely on the internal structure of the model but instead evaluate changes in the output when specific inputs are altered, whether it means removed, modified or aggregated.
   - **Permutation Feature Importance** measures the decrease in performance when the values of a particular feature are shuffled, [54], [55].

- **Leave-One-Out (LOO) or Leave One Feature Out (LOFO)** evaluates the importance of individual features by removing them one at a time and measuring the impact on the model's predictions.
- **Perturbation-based** methods work by perturbing features, whether it means removing or modifying, and measure the influence that these perturbations have on the output.
- **SHAP (SHapley Additive exPlanations)** based on game theory, it assigns importance values to each feature considering all possible subsets of features to estimate their contribution. [17].

3. **Visualization-Based Methods**: These techniques represent the effect of features in a visual format, making it easier for the user to understand the behavior of the model.

   - **Partial Dependence Plots (PDPs)** show the average effect of a feature on the predicted outcome, showing if the relation between input and output is linear or more complex [56].
   - **Individual Conditional Expectation (ICE) Plots** improve PDPs by visualizing the effect of a feature at the individual instance level [57].
   - **Saliency** are applicable to any image classifier, showing the pixel that were most relevant for a certain image classification [58]. Saliency is at the base of many other visualization techniques, such as Grad-CAM [47] and SmoothGrad [59].

4. **Rule-Based Explanations**: Some model-agnostic methods generate local rules that approximate the decision process for a specific instance.

   - **Decision lists and sets** extract logical rules from a model to generate explanations in a simple *"if-then"* statement [60]. In case of overlapping rules, in a list the first rule is picked, in a set there is a voting mechanism to decide which rule to apply.
   - **Anchors** provide highly precise *"if-then"* rules that explain the sufficient conditions under which a model makes a certain decision [61].

5. **Instance-Based Explanations**: These methods explain individual predictions, exploring how the model behaves for specific instances rather than analyzing the global patterns.

   - **LIME (Local Interpretable Model-Agnostic Explanations)** approximates the behavior of a complex model around a specific prediction by training a locally interpretable model, such as a linear regression [18].
   - **Counterfactual Explanations** generate a similar instance with slight modifications to understand the minimal change that would lead to a different prediction [62].

6. **Example-Based Explanations**: Similar to instance-based, but instead of analyzing features, these methods select or generate instances (examples) to generate explanations.

   - **Prototypes and criticism** are used to identify representative samples of the input (prototypes) and also what is not accurately represented by prototypes (criticism) [63].
   - **Nearest neighbors** retrieve the most similar case or cases from the training data to explain the decision
   - **Counterfactual explanations**, seen before, can also be categorized here.

We will now dive deeper into the details of some of these methods used to generate explanations for the scope of this thesis.

4.2 Leave-One-Out (LOO) or Leave-One-feature-Out (LOFO)

The Leave-One-Out (LOO) method is a model-agnostic perturbation-based technique used to evaluate the contribution of each feature to the prediction of a model. To do so, features are **removed or perturbed**, one by one from the input, then the modified input is fed to the model and measuring the impact of this removal based on the **difference in output probabilities**.
Removing, especially in the case of audio data, can introduce differences in the audio duration and this can cause problems in the downstream process, so a zeroing perturbation would be preferred. The perturbation can be of any type, such as zeroing the input, using the mean value, or, in the case of speech, introducing different types of noise as explained in section 4.5. This technique is based on the assumption that a big change in the prediction probabilities when a feature is removed is associated to a high level of importance for that feature, while a small change means that the feature is not important for that target.

Mathematically , we indicate with $x \in \mathbb{R}^n$ the audio signal and $\{x_1, ..., x_n\}$ the set of features (e.g. words or phonemes) that the audio is segmented with.
For a classification model f applied for a task such as intent classification or emotion recognition, the output probabilities are $f(y = k|x)$ for a target k given the input x. We can define the relevance, meaning the importance, $r(x_i) \in \mathbb{R}$ of each feature x_i to the model's prediction for a target class k as:

$$r(x_i) = f(y = k \mid \mathbf{x}) - f(y = k \mid \mathbf{x} \setminus x_i) \tag{4.1}$$

where $x \setminus x_i$ is the signal x without the feature x_i.
The higher the absolute value of $r(x_i)$, the more impact the segment that was removed

from the prediction has. If the difference is positive, it means that the feature x_i contributes to the probability of belonging to class k, while a negative result means that x_i points toward another other class. If the classification is binary, it pushes towards the other class, if it is multi-class, it's any other possible class [28], [64].

The advantages of LOO are the following:

- **Simplicity** of the concept. It directly connects the absence of a feature to the change in the model's prediction and its importance, making the explanation easy to interpret and visualize.
- **Model-agnostic** design, making it applicable to any black-box model with minimum adaptations.
- **Local explanations** obtained by evaluating the impact of each feature for a given input, particularly useful for application requiring instance-based interpretation.
- **Good baseline** for other more sophisticated models to build on, like SHAP or LIME.

The limitations of LOO instead are:

- **Computations can ramp up quickly** for dataset with a fine-grained feature segmentation, requiring many computations for the same input. For i features, it requires $i + 1$ evaluations.
- **Ignores feature interactions** since it evaluate each feature independently and can create unrealistic data. This can cause problems when evaluating inputs with features that interact with each other, which leads to misleading importance scores.
- **Non additive** approach can be problematic when explaining a prediction that is a sum of individual contributions. Similar is the case of phoneme-level segmentation, where a single phoneme can carry little importance but when two or more phonemes are aggregated, their importance can drastically increase.
- **Simplicity** can be a good thing, but in case we need a deeper understanding we have to choose a different approach.

4.3 Local Interpretable Model-Agnostic Explanations (LIME)

LIME is part of the instance-based explanation methods and is defined as a **local surrogate** interpretable model. Surrogate models are trained to imitate the output predictions of the black-box model, but this type only focuses on individual predictions

instead of trying to imitate the global model.

The idea behind LIME is to use the black-box model as an oracle that you can question how many times you want and need to understand why it made a certain decision. LIME tests what happens when we feed to the model many slightly different inputs based on the original one. It generates a new dataset consisting of perturbed samples and the obtained prediction of the black-box. The new dataset is used to train an interpretable model, with weights based on the proximity of the new samples to the original one. This model respect the property of **local fidelity**, meaning it is a good local approximation, but this does not imply global fidelity.
Similarly to LOO, the perturbation can be zeroing the input, using a mean value or noises. The difference is that each of these perturbed samples can have multiple segments masked, capturing the effects of multiple missing pieces.
To get an importance score for each of the original features we leverage the coefficient of the interpretable model. The scores we derive from LIME have the same meaning of LOO as well, with higher score indicating higher relevance and lower score pointing towards another class.

Mathematically, a local surrogate model is defined as follows:

$$\text{explanation}(x) = \arg\min_{g \in G} L(f, g, \pi_x) + \Omega(g) \tag{4.2}$$

Where, x is the instance we want to explain, g is the model that minimizes the loss L between all possible interpretable models G, while $\Omega(g)$ is the model complexity that we want to keep low (e.g. few features). The loss L measures how close the explanation is to the one of the original model f, while the locality is measured with π_x, meaning how much a perturbed sample can differ from x. The loss $L(f, g, \pi_x)$ defines the concept of **locality**, while $\Omega(x)$ defines that of **interpretability**. This equation translate to: *"The explanation of an instance x, is the model g that minimizes the loss L while respecting the complexity constraints $\Omega(g)$ in the π_x locality"*.

The advantages of LIME are the following:

- **Model-agnostic** design, reusable even for different underlying models.
- **Adaptable representation** of the explanation based on who is going to consume it, with the possibility to use a different representation from how the black-box model represents the input.
- **Local and short explanations** obtained through the surrogate model and controlled by the number of features selected.
- **Works for multiple input types**, tabular, text and images.
- **Feature attributions** are provided by the model.

The limitations and disadvantages instead are:

- **Ignores features interaction** since perturbations are produced from a Gaussian distribution, and this can lead to unrealistic data samples that are then used to learn local explanations and misleading importance scores.
- **Complexity has to be defined in advance** by the user, who has to choose neighborhood locality and the number of perturbed samples.
- **Instability of the explanations**, which means that on multiple runs the explanation for two very similar inputs can vary greatly ([65] & [66]), because the neighborhood depends on random perturbations. This has also repercussions on the trust we can put on the explanations.
- **It's possible to manipulate them** to hide biases in the black-box model [66] & [67] .

4.4 SHapley Additive exPlanations (SHAP)

SHAP is a local method that aims to explain the prediction of a single instance x through the computation of the contribution of each feature to the final prediction. It is based on the computation of the Shapley values, which will be the first focus of this section. Then, we will see how the SHAP method uses this concept and focus on the KernelSHAP variation [17]

4.4.1 Shapley Values

The explanation methods that rely on the removal mechanism often fail to consider the interaction that features can have. This can lead to results that are misleading or samples that are not realistic. To solve this, we need an approach that is able **consider the contribution of multiple features**: we do not want to measure the effect of removing feature x_i by itself, but rather consider its impact when removed in combination with other features (e.g. x_i and x_j, or more). A possible solution to aggregate these importance scores is using Shapley values.

Shapley values calculation comes from the game theory, where each player is assumed to collaborate with the others, and that allows us to assign to each of them a contribution score [68]. In linear models or regressions it is easy to assign a score, based on the coefficient each component has times the value of that component. In black-box models, this is not possible and we need more complex solutions, as LIME or SHAP.
Shapley values are easier to understand with an example, before diving into their math.

Example

Suppose there is a team of people working on a project. Usually, the combination of two or more people is more productive than the sole contribution of a single person, but

this also depends on how effective these people work with each other. In a group of three people, $(A,\ B,\ C)$, A might work well with B but not as much with C, and to achieve the completion of the job they have to work together in sequence.

The idea behind Shapley values is to form a group by **adding people one-by-one** and see how much the output changes. Let's say that if A works by themselves, they achieve 10 points, B achieves 20 points and C achieves 30 points. In the case of two people working together, we might have $(A + B)$ getting to 60 points because their combination work well together, while $(A+C)$ only 50 and $(B+C)$ get to 65. We can also consider the different sequences in which $A,\ B,\ C$ can work. For example, A starts, then B joins and finally C joins, translating to an output increase of $0 \rightarrow 10$, then $10 \rightarrow 60$ and $60 \rightarrow 100$. Another sequence could be $C \rightarrow B \rightarrow A$, or any other. These allows us to compute the contributions of each person, for the first sequence $A = 10,\ B = 50,\ C = 40$ and the other sequence $A = 35,\ B = 35,\ C = 30$ and so on. Taking the average of their contribution across all sequences for each person gives the Shapley values of this problem:

$$A = (10 + 50 + 35 + 10 + 50 + 40)/6 = 195/6 = 32.5 \tag{4.3}$$

$$B = (50 + 20 + 35 + 35 + 20 + 20)/6 = 180/6 = 30 \tag{4.4}$$

$$C = (40 + 30 + 30 + 55 + 30 + 40)/6 = 225/6 = 37.5 \tag{4.5}$$

These values represent the average contribution of each person to the job, considering all possible interactions with each other.

Properties

Before moving to the mathematical formulation of Shapley values, we will focus on the properties that this method respect, along with their formulas to anticipate some concepts for later. It's important to note that Shapley values is the only attribution method that satisfies all of the following:

1. **Efficiency**: All the features contributions must add up to 100% of the value of the whole team. The formulation is:

$$\sum_{i \in N} \phi_i(v) = v(N) \tag{4.6}$$

   where $\phi_i(v)$ is the contribution of player i for a certain coalition.

2. **Symmetry**: If two features have the same marginal contributions to all the possible coalitions, they should have the same contribution value.

$$\text{If } v(S \cup \{i\}) = v(S \cup \{j\}) \text{ for all } S \subseteq N\{i, j\},\ \text{then } \phi_i(v) = \phi_j(v) \tag{4.7}$$

   Where $v(S)$ is the function that predict the contributions for the features in set S.

3. **Null-player**: A features that does not change the prediction, regardless of its coalition, should have Shapley value of 0.

$$v(S \cup \{i\}) = v(S) \text{ for all } S \text{ then } \phi_i(v) = 0 \tag{4.8}$$

4. **Linearity**: If a game is the result of two or more functions v and w, then the resulting function is the sum of the contribution from v and w:

$$\phi_i(v + w) = \phi_i(v) + \phi_i(w) \tag{4.9}$$

Mathematical formulation

Intuitively, the Shapley value of a feature j is the average marginal contribution ϕ_j across all possible combinations to the prediction of a particular instance (Equation 4.10). This does **not** mean that it is the difference in prediction when we remove the feature, but instead it is the **average** contribution of a feature to the prediction, through different coalitions.

For any subset S of features that does not include j, the **marginal contribution** of feature j is defined as:

$$\Delta f_j(S) = f(S \cup \{j\}) - f(S) \tag{4.10}$$

The **Shapley value** for feature j is calculated as the weighted average of these marginal contribution, so all the possible feature combinations, and the contribution of j is defined as:

$$\phi_j = \sum_{S \subseteq N \setminus \{j\}} \frac{|S|!(|N| - |S| - 1)!}{|N|!} \Big[f(S \cup \{j\}) - f(S) \Big] \tag{4.11}$$

where:

- N is the set of all features.
- S represents a subset of features not including j.
- $f(S)$ is the prediction (or payoff) when considering the features in S.

Even though the Shapley value allow to consider for the interaction between features, the computation time for this method is exponential with the number of features, and the exact computation can become extremely intensive. This is why it's usually preferred to use an approximation method such as the Monte-Carlo sampling introduced in [69] which samples random coalitions, or the two versions of SHAP introduced in [17].

4.4.2 SHAP

SHAP introduces a new way to compute the estimation of Shapley values, with two specific methods called **KernelSHAP**, as model-agnostic approach, or TreeSHAP, specifically for tree-based models [17]. On top of that, it proposes a way to aggregate local explanations to achieve global insights of the underlying black-box model. We will focus on the KernelSHAP variation because of its ability to explain any model.

Based on Shapley values and game theory, SHAP's goal is to explain a prediction for an instance based on how much each feature has contributed to that prediction. Each feature (or group of features, such as grouped pixels) acts as a player and SHAP. Differently from Shapley values, in SHAP the explanation is represented as a linear model, an additive

feature attribution method, bridging LIME and Shapley values.
An **additive feature attribution method** is defined as follows:

$$g(x') = \phi_0 + \sum_{j=1}^{M} \phi_j x'_j \tag{4.12}$$

where:

- g is the surrogate interpretable model of the original model f
- x' are the simplified features derived from z
- ϕ_0 is the average prediction taken as a baseline ($\phi_0 = \mathbb{E}[f(X)]$
- ϕ_j is the contribution of the simplified $j - th$ feature
- M is the number of simplified features

The simplified features derived from z are mapped through a function h where $z = h(x')$, and it's usually a binary indicator ($x'_j \in \{0, 1\}$ of the presence (1) or absence (0) of a feature j. M instead can be as many as the original features, but it can also be less in case of aggregations, or more in case of segmentation. In other terms, we want an interpretable model g that explains the original model f through a simplified feature representation x', so we want $g(x') \simeq f(x)$.

Properties

Aside from the underlying properties of the Shapley value, that SHAP also satisfies, there are three main properties that SHAP imposes on the additive surrogate models:

1. **Local Accuracy**: The model should match the original model's prediction when all features are present, $g(x') = f(x)$.
   In many formulations, $\phi_0 = \mathbb{E}[f(X)]$ which ensures that at the point x the surrogate is faithful to f:

$$f(x) = g(x') = \phi_0 + \sum_{i=1}^{M} \phi_i x'_i = \mathbb{E}[f(X)] + \sum_{i=1}^{M} \phi_i x'_i \tag{4.13}$$

2. **Missingness**: If a feature is absent, meaning it's missing from the input feature coalition, it gets an attribution of 0.

$$x'_i = 0 \quad \Longrightarrow \quad \phi_i = 0. \tag{4.14}$$

3. **Consistency**: If a model f changes so that the marginal contribution of a feature i stays the same or increases, then the corresponding score ϕ_i should not decrease either.

Let $f_x(z') = f(h_x(z'))$ and $z' \setminus i$ indicates that $z'_i = 0$ (feature i is absent). For any two models f and f', if:

$$f'_x(z') - f'_x(z' \setminus i) \geq f_x(z') - f_x(z' \setminus i)$$

$$\forall z' \in \{0,1\}^M$$

then

$$\phi_i(f', x) \geq \phi_i(f, x)$$

From this property automatically follow the linearity, null-value and symmetry properties of the Shapley values.

Connecting to Shapley values

The only additive feature attribution method (Equation 4.12) that satisfies all of these properties is the set of Shapley values. In this case, the different coalitions are represented by binary vectors z', indicating which features are present or absent. The formula for the Shapley value of a feature i for a model f and an instance x becomes:

$$\phi_i(f, x) = \sum_{z' \subseteq M} \frac{(|M| - |z'| - 1)!\,|z'|!}{|M|!} \Big(f_x(z') - f_x(z' \setminus \{i\}) \Big) \tag{4.15}$$

in which $|z'|$ represent the number of features present in z' and $z' \setminus i$ the same set with feature i set to 0.
In practice, obtaining these values remains very complex and this is why **SHAP introduces an approximation** to compute them (KernelSHAP & TreeSHAP), to make the model applicable in more practical case.

In this framework, $f_x(z')$ means evaluating the original model f on a version of x where only a subset of features of z' is present, the non-zero entries, which we call S. This is written as:

$$f_x(z') = f(h_x(z'))$$

where h is the mapping function from z' to the original input for f. When features are missing, SHAP approximates their values by taking the conditional expectation of the model output over the missing features:

$$f(h_x(z')) = \mathbb{E}[f(z) \mid (z)_s]$$

where z_S indicates the values of the features in S. By integrating over the distribution of the missing features, SHAP can simulate the effect of their absence, by assuming feature independence to simplify this calculation and $\bar{S}$ being the complement of S:

$$\mathbb{E}[f(z) \mid (z)_s] \approx f((z)_s, \mathbb{E}[z_{\bar{s}}])$$

As previously seen, each ϕ_i represents how much each feature shifts the prediction from the baseline:

$$\underbrace{\mathbb{E}[f(z)]}_{\phi_0} \longrightarrow \phi_1 \longrightarrow \phi_2 \longrightarrow \ldots \longrightarrow \phi_M = f(x)$$

Considering each ϕ_i as the weighted average contribution of all possible paths in which a feature can appear.

KernelSHAP

Since the exact computation of Shapley values is exponential in the number of features M, KernelSHAP tries to address this problem by **sampling coalitions of features** and fitting a weighted linear model to approximate the underlying Shapley values. This method is also model-agnostic because it only requires to query the black-box model f for specific instances.
KernelSHAP can be seen as a refinement of LIME, because similarly it minimizes a loss function L that measures how well the surrogate mode g approximates a function f in the neighborhood of x. In LIME it also appears a regularization term $\Omega(g)$ for interpretability, while for KernelSHAP this constraint is dropped and set to $\Omega(g) = 0$, meaning no penalty for interpretability.
The locality is defined inside the loss function:

$$L(f, g, \pi_x) = \sum_{z' \in \mathcal{Z}} \pi_x(z') \left(f(h_x(z')) - g(z')\right)^2 \tag{4.16}$$

particularly,

$$\pi_{x'}(z') = \frac{M - 1}{\binom{M}{|z'|} |z'| \left(M - |z'|\right)} \tag{4.17}$$

where $|z'|$ is the number of non-zero elements in z and M the maximum size of a coalition. This kernel pushes towards sampling coalitions of different sizes and it corrects based on how often each size appears.

Pseudocode In practice, what KernelSHAP does is:

1. Samples K coalitions $\{z'_{k,j}\}$, where $z'_{k,j} = 1$ if the feature j is present
2. Computes $f(h_x(z'_k))$, meaning the prediction of the model f after converting the instance z'_k to the original feature space
3. Assigns a weight to each coalition z'_k with SHAP kernel, similarly to Equation 4.17:

$$\pi_x(z'_k) = \frac{M - 1}{\binom{M}{|z'_k|} |z'_k| \left(M - |z'_k|\right)}$$

4. Fits a weighted linear model

$$min\ L(f, g, \pi_x) = \sum_{z' \in \mathcal{Z}} \pi_x(z') \left(f(h_x(z')) - g(z')\right)^2$$

where

$$g(z') = \phi_0 + \sum_{i=1}^{M} \phi_i z'_i$$

and the coefficient of the linear model ϕ_i are the approximated Shapley values

Global insights

Even though SHAP is a local explanation method, there are ways to derive global insights for an entire dataset. This can be obtained by aggregating each Shapley value to identify which features are most influential, on average.

Global Feature Importance is an approach that computes the **average of the absolute Shapley values** for each feature across all dataset.

1. Compute SHAP for each instance of the dataset, obtaining the importance of each feature for all the instances.
2. Compute the average of the absolute Shapley values per feature:

$$I_j = \frac{1}{n} \sum_{i=1}^{n} |\phi_j^{(i)}| \tag{4.18}$$

where $\phi_j^{(i)}$ is the SHAP value for feature j in instance i, and n the total number of instances.
The higher the value of I_j, the stronger the impact on the *overall* model's prediction is. Figure 4.7 shows how to represent these averages through a bar chart for a multi-class model.

SHAP Beeswarm Plot is a visualization tool also called summary plot, in which each point of the plot corresponds to the SHAP value of a feature for a particular instance, where the x-axis shows the SHAP value and on the y-axis the feature name sorted by the sum of the SHAP value across all samples (Figure 4.8).

Dependence Plot offer a way to visualize feature interactions. For a specific feature j we plot $(x_j^{(i)}, \phi_j^{(i)}$, for each instance i, having on the x-axis the feature value and on the y-axis the corresponding SHAP value. Points can be colored based on another feature k to visualize potential interactions (Figure 4.9).

Advantages and Limitations

The advantages of SHAP

- **Model-agnostic & local explanations**: SHAP can be applied to any model by sampling feature subsets and the approximation of Shapley values.

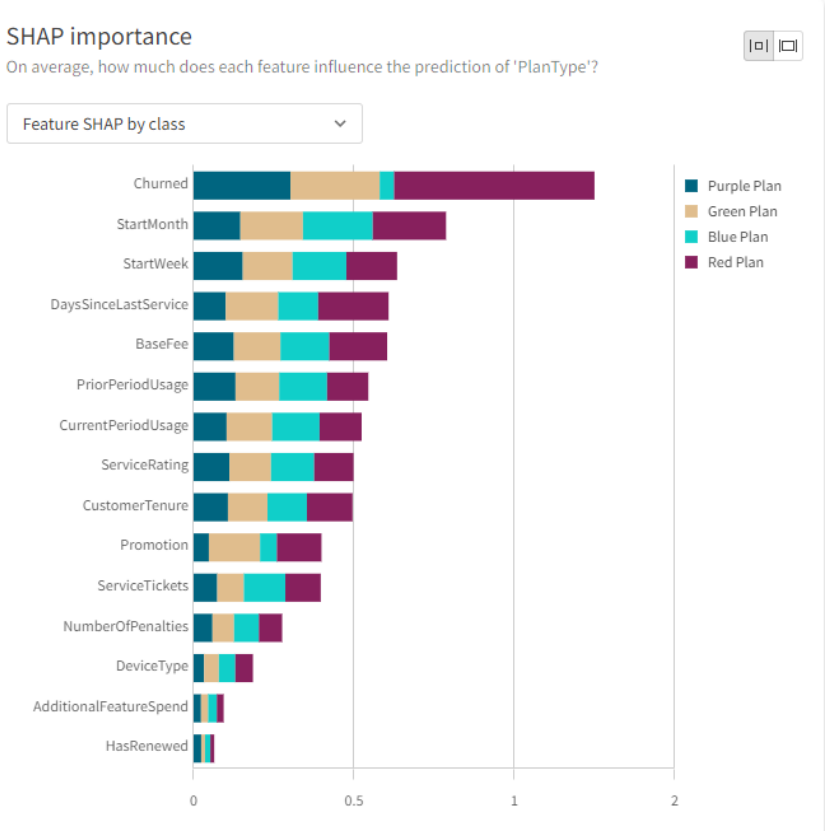

Figure 4.7. Global insights of a multi-class model, showing feature importance for each possible class output.

- **Global interpretability**: Through the aggregation of local explanations it's possible to derive global insights.
- **Interaction among features**: By averaging over all possible coalitions, SHAP can account for the possible feature interactions.
- **Reduced computational complexity**: Thanks to approximations and well-implemented libraries across multiple languages.
- **Theoretical foundation**: SHAP inherits the game-theory properties of Shapley values (efficiency, symmetry, null player, linearity), as well as adding some like consistency, missingness, and local accuracy.

The limitations of SHAP

- **Requires data access**: SHAP needs data access to compute $f(h_x(z'))$
- **Independence assumption**: To simulate the absence of features, it assumes that features are independent.

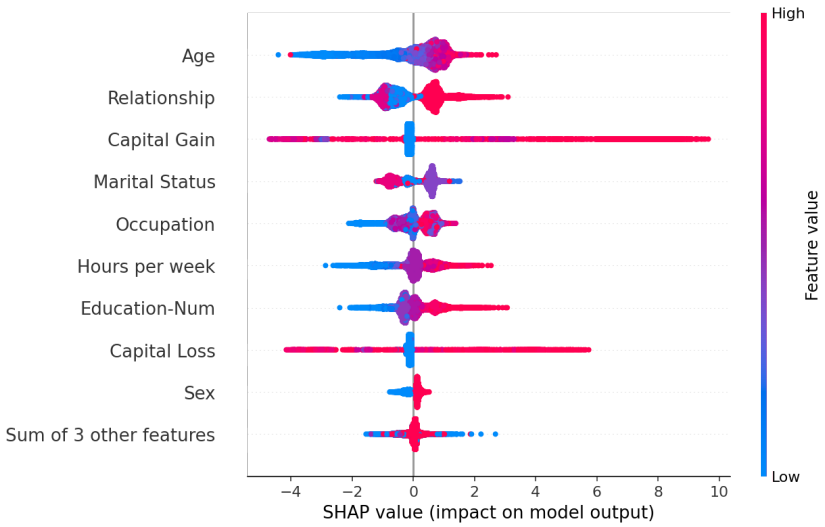

Figure 4.8. Global insights obtained from a beeswarm plot.

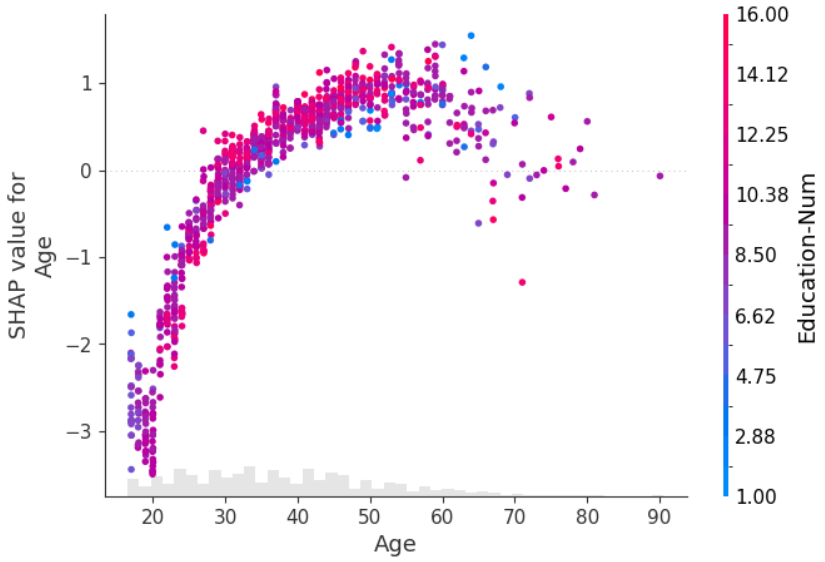

Figure 4.9. Feature dependence for the feature *Age* and the feature *Education_Num*.

- **Unrealistic data samples**: The absence of feature and independence assumption can create unrealistic samples, that can make the explanations less accurate.

- **Approximation**: KernelSHAP is an approximation of the true Shapley values, and is based on random instances which can also cause variance in the estimations.

- **KernelSHAP is slow**: In the case of large number of features, KernelSHAP can become slow. Other methods exists to speed up the process, such as new sampling strategies [70], or other approximation methods such as FastSHAP [71], Gradient-based SHAP, or model-specific SHAP methods such as TreeSHAP and LinearSHAP.

TreeSHAP

TreeSHAP is a specialized variant of SHAP, which is tailored to tree-based models such as random forests and gradient-boosted trees [51]. It leverages the structure of the decision trees to compute the exact Shapley value in polynomial time with respect to the number of leaves. It is significantly faster than KernelSHAP for large trees ensembles and has the same theoretical foundation that Kernel guarantees.

4.5 Paralinguistic Attributions

Aside from the semantic information that we can convey with words, speech also includes paralinguistic information. These can carry additional information, focusing on **how things are said**, rather then what is said. Paralinguistic features can be attributed to the speaker's voice, such as pitch, speed and rhythm or external condition as the background noise.
This type of information is important to capture in the field of ASR, since:

- It can lead to a more **holistic understanding of speech**, capturing the different ways in which a sentence can be delivered.
- It is used in **emotion recognition** since paralinguistic features are the main carrier of these information.
- It can help the **speaker identification**.
- In certain languages the same word **can change meaning** if spoken with different attributions.

This can also help XAI methods, because it can give **insights into the non-verbal dimension of speech** and how it affects model predictions, which can be used to identify biases and weaknesses of the models. Finally, it can help in building models that are more resistant to variations, whether in pitch or in background noises.

To tackle this problem, we can use **perturbation-based techniques** to attribute importance to different speech features, which in this case are represented by the paralinguistic attribute we choose.

Mathematically We indicate with $p := f : \mathbb{R}^n \to \mathbb{R}$ a function to extract a paralinguistic measure of interest, such as the pitch, and maps the input of an audio signal to a real value measure $p(x)$. We then transform x into $\widetilde{x}$ such that $p(\widetilde{x})$ is either higher or lower than $p(x)$. In other words, this means we *shift the pitch* up.

To compute the effect of p on the prediction of a certain objective class k, we perturb x multiple times and average the result as:

$$r_p(\widetilde{x}) = f(y = k|x) - f(y = k|\widetilde{x}) \tag{4.19}$$

$$r(x, p) = \frac{1}{|\widetilde{X}_p|} \sum_{\widetilde{x} \in \widetilde{X}_p} r_p(\widetilde{x}) \tag{4.20}$$

where $r_p(\widetilde{x})$ is the *effect* of a single perturbation, $\widetilde{X}_p = \{\widetilde{x}_1, \ldots, \widetilde{x}_t\}$ is the *set* of t perturbed audio signals along the p feature, t depends on the granularity of the perturbations applied.
$r(x, p)$ is the relevance, measuring how sensitive the model's decisions are to the feature p. This value is bound to $[-1, +1]$, specifically if:

- $r(x, p) > 0$, then increasing $p(x)$ *lowers* the probability of class k
- $r(x, p) < 0$, then increasing $p(x)$ *raises* the probability of class k
- $r(x, p) \approx 0$, then the model is robust to changes of $p(x)$

4.6 Implementation of Interpretable Techniques

In this section we will see how the previously introduced methods have been implemented, studying a summary of their headers containing the possible parametrization and functionality of each method. Moreover, some under-the-hood functions are described for better clarity of what happens when these methods are called, as well as the parametrization for some important calculations (section 4.6).
All the code can be used from the `speech_xai__exp.ipyb` notebook. To pick a specific method to create an explanation, we can use the following syntax:

```
explanation = benchmark.explain(
    audio_path=audio_path,
    methodology='LOO')
```

In this case an explanation for the audio corresponding to the `audio_path` with the `LOO` methodology with default parameters will be returned.

The full code is available at (`https://github.com/Raoolo/Phoneme-Thesis`). We will also see the main frameworks and libraries used to support the code, such as WhisperX and others. Through the combination of the parameters presented for each of the explainability methods, we can alter the explanations and get more insights from the same audio sample.

In Figure 4.10 we can see how the program achieves a result through a graphic, to easily understand how the data flows from the input layer to the output.

Once the data is loaded and in its raw form, the first step of the pipeline is to obtain a baseline classification from the model. Then, the audio is transcribed, and at the same time the transcription is converted to its phonemic version. Once we have this data, we can perform phoneme alignment. This step splits the transcription into its phonetic

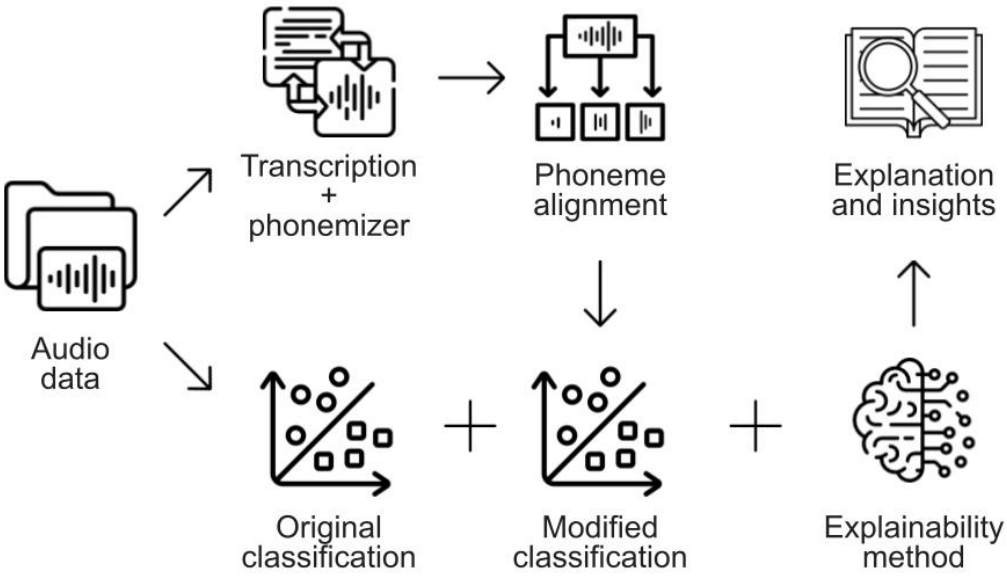

Figure 4.10. Graph showing the main steps of how the program achieves explanations.

units and provides the timestamp of each unit, allowing the program to distinguish each phoneme based on its start and end time. Once we have this information, we can utilize an explanation method, that, after perturbing the audio using the phoneme timestamps, obtains the new classification scores. Comparing the original classification scores and the new ones, the method returns an explanation and we are able to extract insights and visualize it properly.

4.6.1 Leave-One-Out (LOO)

This method evaluates the contribution of each feature in an audio by removing them one at a time and then quantifying the change in the model's output. Through this method it's also possible to perturb the features with different modifications, such as shifting their pitch, to see how it affects the predictions. The full implementation is found inside the *explainers* folder, `loo_speech_explainer.py`.
The simplified workflow is as follows:

1. **Load audio**: Load the audio using pydub [72].
2. **Transcription**: Obtain the transcription of the audio file using *WhisperX* [73] and obtain the details about each feature. Phonemes are obtained thanks to a *Wav2Vec* model [7] fine-tuned for phonetic recognition [74].
3. **Feature modification**: For each feature, create a new audio in which the feature is either removed or perturbed with *audiomentations* [75].

4. **Generating explanations**: Obtain the predictions for each new audio and compare them with the original prediction to obtain the importance scores of each feature.

Parameters The following are the parameters that modify the functionality of this class:

- `audio_path`: a *string* that corresponds to the **audio path** that contains the *.wav* file.
- `removal_type`: a *string* that corresponds to the type of **removal** we want to apply to the feature we are removing. This parameter is also used in the case we want to **perturb** a feature instead of removing it.
  - To remove an audio, we can use `"nothing"` or `silence`. By default it is set to `"nothing"`.
  - To perturb an audio, we can use any of the following modifications: `"pitch shifting"`, `"pitch shifting down"`, `"pitch shifting up"`, `"reverberation"`, `"time stretching"`,`"time stretching down"`, `"time stretching up"`, `"stress"`, `"degradation"`, `"noise"`, `"white noise"`, `"pink noise"`, `"claps"`, `"intensity"`
- `phonemization`: a *boolean* value that activates the **phoneme-level explanation**. If not provided, or set to *false*, the method will work at word-level.
- `window_size`: an *integer* that turns on the **aggregation** of two or more phonemes into groups of size *window_size*. By default this is off.
- `respect_word_boundaries`: a *boolean* value subordinated to the `window_size` parameter. In case of aggregation, this avoids the aggregation of phonemes from **different words** (e.g. the last phoneme of the preceding and the first phoneme of the following). By default this is on (set to *true).*
- `sliding`: a *boolean* value subordinated to the `window_size` parameter. It is used to choose between two different types of **sliding windows**. By default this is set to *false*.
  Using characters instead of phonemes for simplicity, when set to *true* for `window_size=3` the aggregations will be *"bed edr dro oom"* rather than *"bed roo m"*. This concept is detailed in subsection 4.6.4.
- `single_perturbation_value`: a *numerical* value that allows the user to study in details the effects of a **single perturbation** value. The value chosen should be appropriate for the perturbation (`removal_type`) chosen.
- `complete_perturbation`: a *boolean* value used to perturb each of the feature with **multiple perturbation** values. By default this is set to *false*.
- `perturbation_list`: a *list* of **multiple perturbations** subordinated to the activation of the `complete_perturbation` parameter.

- `words_transcript`: a *list* containing the **words of the transcription** of the audio. If not provided, the model will automatically create a transcription.
- `visualization`: a *boolean* values used to **activate the visualization** of graphs.
- `display_audio`: a *boolean* value used for **logging and debugging**. Set to *false* by default
- `verbose` and `verbose_target`: a *boolean* and a subordinated *integer* value used for **logging and debugging** a specific class of interest. By default set to *false*.

4.6.2 Local Interpretable Model-Agnostic Explanations (LIME)

This method uses the LIME framework to locally approximate the behavior of an audio black-box model. Instead of solely removing features, LIME creates a neighborhood of perturbed audio samples and fits a surrogate interpretable model to estimate the contribution of each feature. The full implementation is found within the *explainers* folder in `lime_speech_explainer.py` and `lime_timeseries.py`.
The simplified workflow is as follows:

1. **Load audio**: Load the audio using pydub [72].
2. **Transcription**: Obtain the transcription of the audio file using WhisperX [73] and obtain the details about each feature. Optionally, phonemes are obtained thanks to a Wav2Vec model [7] fine-tuned for phonetic recognition [74].
3. **Interpretable representation**: LIME requires the input to be represented with interpretable features, in this case each phoneme is represented with a binary mask that indicates whether it is present (1) or absent (0) [29] [30].
4. **Neighborhood generation**: Create a set of perturbed audio samples by randomly deactivating features.
5. **Surrogate model**: Using an adapted version of the lime library [28], fit a weighted linear model on the perturbed samples.
6. **Generate explanations**: Based on the regression coefficients, derive the importance scores and build the explanation.

Parameters The following are the parameters that modify the functionality of this class:

- `audio_path`: a *string* that corresponds to the **audio path** that contains the *.wav* file.
- `removal_type`: a *string* that corresponds to the type of **removal** we want to apply to the feature we are removing. In this case, the supported removal types re `"silence"`, `"noise"`, `"mean"`, and `"total_mean"`. By default it is set to `"silence"`.

- `num_samples`: an *integer* that represents the **number of perturbed samples** to generate to train the surrogate model.
- `phonemization`: a *boolean* value that activates the **phoneme-level explanation**. If not provided, or set to *false*, the method will work at word-level.
- `window_size`: an *integer* that turns on the **aggregation** of two or more phonemes into groups of size *window_size*. By default this is off.
- `respect_word_boundaries`: a *boolean* value subordinated to the `window_size` parameter. In case of aggregation, this avoids the aggregation of phonemes from **different words** (e.g. the last phoneme of the preceding and the first phoneme of the following). By default this is on (set to *true)*.
- `sliding`: a *boolean* value subordinated to the `window_size` parameter. It is used to choose between two different types of **sliding windows**. By default this is set to *false*.
  Using characters instead of phonemes for simplicity, when set to *true* for `window_size=3` the aggregations will be *"bed edr dro oom"* rather than *"bed roo m"*. This concept is detailed in subsection 4.6.4.
- `words_transcript`: a *list* containing the **words of the transcription** of the audio. If not provided, the model will automatically create a transcription.
- `visualization`: a *boolean* values used to **activate the visualization** of graphs.

4.6.3 SHapley Additive ExPlanations (SHAP)

This method uses SHAP values to quantify the contribution of each feature to the model's prediction. SHAP computes importance scores by comparing the model's output on the original audio with outputs on perturbed versions where certain features are masked. The full implementation is located in the *explainers* folder inside the file `shap_speech_explainer.py`.
The simplified workflow is as follows:

- **Load Audio**: Load the audio using pydub [72].
- **Transcription**: Obtain the transcription of the audio file using WhisperX [73] and obtain the details about each feature. Optionally, phonemes are obtained thanks to a Wav2Vec model [7] fine-tuned for phonetic recognition [74].
- **Feature masking**: Create an interpretable representation of the features, in this case a binary 0,1 mask, where 0 means absent and 1 means present.
- **Background data**: Create a set of binary samples with various combination of presence and absence of the features.
- **SHAP value computation**: Using the `KernelExplainer` from *shap* [17], we compute the SHAP values for our instance.

- **Generate explanations**: Map the SHAP values to the features and generate the corresponding explanation.

Parameters The following are the parameters that modify the functionality of this class:

1. `audio_path`: a *string* that corresponds to the **audio path** that contains the *.wav* file.
2. `removal_type`: a *string* that corresponds to the type of **removal** we want to apply to the feature we are removing. In this case, the supported removal types re `"silence"`, `"nothing"`. By default it is set to `"nothing"`.
3. `num_samples`: an *integer* that represents the **number of background samples** to generate used in SHAP.
4. `phonemization`: a *boolean* value that activates the **phoneme-level explanation**. If not provided, or set to *false*, the method will work at word-level.
5. `window_size`: an *integer* that turns on the **aggregation** of two or more phonemes into groups of size *window_size*. By default this is off.
6. `respect_word_boundaries`: a *boolean* value subordinated to the `window_size` parameter. In case of aggregation, this avoids the aggregation of phonemes from **different words** (e.g. the last phoneme of the preceding and the first phoneme of the following). By default this is on (set to *true)*.
7. `sliding`: a *boolean* value subordinated to the `window_size` parameter. It is used to choose between two different types of **sliding windows**. By default this is set to *false*.
   Using characters instead of phonemes for simplicity, when set to *true* for `window_size=3` the aggregations will be *"bed edr dro oom"* rather than *"bed roo m"*. This concept is detailed in subsection 4.6.4.
8. `words_transcript`: a *list* containing the **words of the transcription** of the audio. If not provided, the model will automatically create a transcription.
9. `visualization`: a *boolean* values used to **activate the visualization** of graphs.

4.6.4 Transcription and Aggregation

The program's transcriptions are based on WhisperX [73], which is a state-of-the-art multi-lingual alignment and transcription model, used to transcribe the audios and obtain the timestamp at different level of granularity.

WhisperX is based on Whisper [8], which is one of the first and most used large-scale speech recognition models across different domains and languages. These models however are prone to inaccuracies, especially when it comes to predicting timestamps. To overcome these challenges, WhisperX was introduced as a time-accurate speech recognition

system, supporting word-level timestamps through forced phoneme alignment and parallel computation to speed up the processing.
Focusing on the **forced phoneme alignment**, this is where WhisperX stands out. After computing the transcription, an external phoneme recognition model is used in the forced alignment step. Under the hood, the system employs Dynamic Time Warping (DTW) to determine the mapping between audio and the text, properly assigning start and end timestamps to each word, and if necessary also to phonemes. In this case, the phoneme recognition model chosen is different from the default one. To achieve even more precise phoneme timestamps for multiple languages, the *facebook/wav2vec2-lv-60-espeak-cv-ft* model has been used [74]. It is based on *Wav2Vec2.0* [7] and fine-tuned on the task of phoneme timestamp recognition. For English utterances a different model can be used, the *Charsiu's phonetic aligner*, which return even more accurate timestamps [38]. Both the model work in a similar, starting from the transcription of the audio they create phonetic representation with a grapheme-to-phoneme approach, and then aligning each phoneme to the audio. This means that they do not take into account the possible different pronunciation of the speaker, but instead are based solely on the transcription provided.
Inside the code, this process can be found in the folder *explainers* in the file *utils_removal.py*. The `transcribe_audio` function is structured as follows:

1. **Load Audio**: Load the audio from `audio_path` using WhisperX integrated function.

2. **Transcription**: Obtain the transcription of the audio file using WhisperX's *large-v3-turbo* model.

3. **Phoneme alignment**: To obtain even more precise phoneme start and end timestamps, we leverage the fine-tuned model *facebook/wav2vec2-lv-60-espeak-cv-ft* or the *Charsiu's phonetic aligner*. The result is aligned using ARPABET or following the International Phonetic Alphabet (IPA) representation (more on this in section 6.1), and structured as follows inside the variable `chars`:

```
'chars': [
    'char': 't', 'start': 0.878, 'end': 0.958, 'score': 0.752,
    'char': 'ɜ', 'start': 0.958, 'end': 1.099, 'score': 0.717,
    'char': 'ː',
    'char': 'n', 'start': 1.099, 'end': 1.159, 'score': 0.664,
    'char': ' ',
    'char': 'ɔ', 'start': 1.159, 'end': 1.32, 'score': 0.879,
    'char': 'f', 'start': 1.32, 'end': 1.4, 'score': 0.992,
]
```

   This is equivalent to the words *"turn off"* in IPA alphabet. Each phoneme also comes with a score, indicating the model's certainty on the predictions.

4. **Removing diacritics**: To clean the structure, we remove the diacritics from it. Diacritics are symbols that are used to modify the sound of a word thus do not have any timestamp associated, such as ː.

5. **Aggregate phonemes**: Optionally, phonemes are also aggregated if `window_size` is set to a number bigger than 1.

   (a) **Index assignment**: Each one of the feature is assigned an index corresponding to the word it belongs to by checking its middle-point timestamp and assigning it to the word that include that time. This information is then added to the previously seen structure of `chars`.

   (b) **Window-based aggregation**: The aggregation is performed using a sliding window approach where phonemes are grouped based on their word index field. The function `aggregate_phonemes` iterates through the list and merges consecutive ones into larger features of size `window_size`.
   Generally, a good size for the window is 2 or 3 phonemes. This is because 2 phonemes can often form a syllable, and 3 phonemes form a syllable or a grapheme, which is one of the smallest constituents in a language, often carrying most of the significance of the word. This aggregation preserves the initial structure of `chars`.

   (c) **Word boundary constraints**: If `respect_word_boundaries` is set to `True`, phonemes are only aggregated within the same word. Otherwise, phonemes can be aggregated across words, leading to a more continuous segmentation. For example, for `window_size=2` the previous structure with the parameter set to `False` would be aggregated as:

   ```
   'chars': [
       'char': 'tɜ', 'start': 0.878, 'end': 1.099, 'score': 0.734,
       'char': 'nɔ', 'start': 1.099, 'end': 1.32, 'score': 0.771,
       'char': 'f', 'start': 1.32, 'end': 1.4, 'score': 0.992,
   ]
   ```

   Effectively merging n and ɔ into a single feature, even from different words.

   (d) **Sliding vs. non-sliding approach**: The aggregation can be performed using either a fixed window or a sliding window. When `sliding` is set to `False`, non-overlapping windows are used, meaning phonemes are grouped into distinct chunks. Instead, with `sliding` set to `True`, a rolling approach is adopted, allowing each phoneme to be included in multiple aggregated units, leading to more granular transitions between phoneme groups. The different result would be, for `windows_size=2`, `respect_word_boundaries=True` and `sliding=True`:

   ```
   'chars': [
       'char': 'tɜ', 'start': 0.878, 'end': 1.099, 'score': 0.734,
       'char': 'ɜn', 'start': 1.099, 'end': 1.32, 'score': 0.691,
       'char': 'ɔf', 'start': 1.32, 'end': 1.4, 'score': 0.936,
   ]
   ```

whereas for `sliding=False`:

```
'chars': [
    'char': 'tɜ', 'start': 0.878, 'end': 1.099, 'score': 0.734,
    'char': 'n', 'start': 1.099, 'end': 1.159, 'score': 0.664,
    'char': 'ɔf', 'start': 1.159, 'end': 1.4, 'score': 0.936,
]
```

By using **different combinations** of these parameters, we can obtain different results and see how each phoneme and its possible aggregation affect the result in different ways.

4.6.5 Evaluation Metrics

To further assess the faithfulness of the previous explanations, we introduce two evaluation metrics: **comprehensiveness** and **sufficiency**. They are based on the *Average Over Perturbation Curve* (AOPC) technique, which quantifies how much a model's prediction changes when parts of the input are either removed or retained, in order of their relevance as estimated by the explanation method.

1. **AOPC Comprehensiveness**: Measures how much the model's confidence decreases when the most important features are *removed*, where a higher score indicates that the removed features were actually important for the decision.

2. **AOPC Sufficiency**: Similarly, measures how much the prediction changes when only the most important features are *retained*. In this case, a lower score indicates that the retained features are sufficient for the model to maintain a similar prediction.

Both of these methods use different importance thresholds to compute their evaluation, and they allow the user to understand if the apparently important features actually matter (comprehensiveness) or if they are enough (sufficiency). By comparing the metrics of the different explainers and evaluators, we can determine which one is able to generate the most faithful explanation.

Chapter 5

Experimental Results

In this chapter, the datasets used for the experiments are described to understand what their aim is and how they are treated for this task (section 5.1).
Finally, we will introduce the results of our experiments and analyze their meaning, studying how different combinations of parameters can change the final results, and why they do it (chapter 6).

5.1 Datasets

Three different datasets have been used for this study. We will shortly introduce them, and see how they are treated so that we can discuss the results in the following chapter 6.

Each dataset has an associated `model_helper` class that serves as a wrapper for the HuggingFace models used for feature extraction and classification [76]. The main functionalities are:

- **Feature extraction**: Preprocesses the raw audio to the expected format for the prediction model. In the code this means squeezing and padding the audio, then transforming it to a tensor.
- **Prediction**: The preprocessed data is passed to the prediction model, which outputs logits (uninterpretable raw values) that are then transformed into probabilities using a softmax.
- **Label mapping**: The output can then be mapped to the corresponding label for more clarity.

5.1.1 Fluent Speech Commands (FSC)

Fluent Speech Commands is a dataset containing *.wav* audio files, each recording contains a single spoken English command usually used in a smart home with a virtual assistant. Such can be *"Turn off the bedroom heat"* or more complex like *"I couldn't hear*

anything, turn up the volume".

FSC is widely used as a benchmark for **intent classification** tasks, specifically each audio has three label slots: *action*, *object*, and *location* and the combination of them forms the *intent*. Each slot takes one of multiple possible values, and some can also be none, such as the location. An intent can be expressed in many forms, this means that `{action: "activate", object: "lights", location: "none"}` can be expressed as *"turn on the lights"*, *"switch the lights on"* and many more. The total of possible different sentences are 248 that can be mapped to 31 intents, including both native and non-native speakers [77].

In the code this dataset is treated using `model_helper_fsc.py` inside the *model_helpers* folder. The prediction model in this case outputs three probabilities, each corresponding to one of the categories above. Specifically for this task, we will use the Wav2Vec 2.0 base [7] fine-tuned with the checkpoint *superb/wav2vec2-base-superb-ic* [78], optimized for intent classification on the FSC dataset.

5.1.2 ITALian Intent Classification (ITALIC) and Speech MASSIVE

Unlike the majority of spoken language understanding datasets that focus on English, the ITALIC dataset is designed for **intent classification** in Italian. Similarly to FSC, it contains *.wav* audio files, corresponding to an Italian utterance, like *"Svegliami alle nove di mattina venerdì"* (*"Wake me up at 9 AM on Friday"*) or *"Spegni le luci per favore"* (*"Turn off the lights please"*). Similarly, the Speech MASSIVE dataset contains the same utterances translated into multiple other languages, such as Spanish, German, French and many more.

These datasets contains audio from a variety of people from different regions and detailed metadata, offering many labeling possibilities such as speaker identification or accent identification, but we will use it only for **single label** intent classification. There are 60 different possible intent labels, covered by thousands different phrases. This means an intent can be expressed through various phrases, for example both *"Controlla se mi funziona il portatile"* (*"Check if my laptop works"*) and *"Mi serve che la posizione sia accesa puoi controllare"* (*"I need the location to be turned on can you check"*) are classified as `"general_quirky"` [79].

In the code this dataset is treated using `model_helper_italic.py` inside the *model_helpers* folder. The model predicts a single label probability, using the multilingual XLS-R [80] and the fine-tuned checkpoints for ITALIC [79]

5.1.3 Interactive Emotional Dyadic Motion Capture (IEMOCAP)

The Interactive Emotional Dyadic Motion Capture dataset is designed for the study of human emotions in speech. It includes 12 hours of data from ten actors engaging in both scripted and improvised dialogues. The dataset includes video, audio and text transcription making it a good foundation for **emotion recognition** [81].

Being used for emotion recognition tasks, each utterance in the dataset is categorized with labels such as anger, happiness, sadness, and others. Multiple utterances can be connected to a label, such as *"I'm really upset about this"* and *"This makes me so angry"* being classified as `"anger"`.

In the code this dataset is treated using `model_helper_er.py` inside the *model_helpers* folder. The model predicts a single label probability, using the Wav2Vec 2.0 base [7] fine-tuned with the checkpoint *superb/wav2vec2-base-superb-er* [78], optimized for emotion recognition on the IEMOCAP dataset.

Chapter 6

Results

In this chapter we will analyze the audios and **explain the outputs** for each of them. This means explaining why the model assigns a certain probability to the predicted label, **providing insights** into how the model produces its predictions. This will be conducted for some audios and with different configurations of the explanation methods, seen in section 4.6.

To analyze them, we will conduct a qualitative manual evaluation, meaning we will manually review the quantitative explanations provided by the methods. This includes evaluating if the explanation is logical and clear, and if it aligns with the human expectations (i.e. **plausibility**, introduced in subsection 4.1.3). Through quantitative faithfulness evaluations methods we will also evaluate the **faithfulness** (introduced in subsection 4.1.3) of the explanation generated.

6.1 FSC Sample Analysis

Following on from the work of [28], which analyzed sentences at the word level, we will now analyze the same sample from the FSC dataset, this time working at different phoneme granularities.
Starting from the utterance with transcription *"Turn up the bedroom heat"* the model correctly predicts the intent as `{action: "increase", object: "heat", location: "bedroom"}`. Table 6.1 shows **word-level explanations** for this sentence for each intent slot, correctly identifying each of them. As expected, *"up"* is associated with the action of increasing, *"heat"* with the object heat and *"bedroom"* with the location. This aligns with what a human would expect from this model, thus making the explanation **plausible** [28].

ARPABET vs IPA When we move to phoneme-level, we do not use the characters composing a word, but instead we have two possibilities to represent phonemes: one is to use the **International Phonetic Alphabet** (IPA) [82], the other is to use **ARPAbet** [83]. The former is an alphabetic notation that is used to represent the sound and intonation of the speech of all languages, while the latter was developed for American-English phonetics and widely adopted in speech recognition models for its simplicity, being only

	Turn	up	the	bedroom	heat
action=increase	0.250	0.545	0.260	0.139	0.021
object=heat	0.000	0.000	0.000	0.014	0.550
location=bedroom	0.002	0.006	0.087	0.997	0.323

Table 6.1. **Word-level explanations with LOO** for the FSC dataset sample, a higher value and deeper color corresponds to more relevance for the prediction.

a subset of the IPA. Both of them have a unique way of identifying phonemes, but it is also possible to map ARPAbet to IPA's sounds as shown in Figure 6.1. For the following examples, we will proceed with ARPAbet. It's also important to note that in this program there is no difference between the chosen representation aside from clarity for the viewer, because the model works end-to-end at the audio level and to apply modifications we only use the start and end timestamps of the features.

ARPAbet	IPA
AA	[ɑ]
AH	[ʌ]
AE	[æ]
EH	[ɛ]
IH	[ɪ]
UH	[ʊ]

Figure 6.1. Conversion between ARPAbet vowels and IPA vowels.

LOO We can start our analysis from the results of the LOO method without phoneme aggregation in Table 6.2 for the same previous sample, *"Turn up the bedroom heat"*.

By inspecting the results, the first thing we have to clear is the representation used. As introduced, phonemes in ARPAbet have a unique way of being represented, either as a single character or as a couple to indicate a particular sound. Based on how the program works, the representation used does not impact the final result, because we only care about their start and end timestamps.
When we look at the predictions, we have to analyze each intent slot and the associated result row. Each cell value indicates the impact on the performance when the corresponding phoneme is removed, where higher values are associated to more significant decrease in the model's ability to predict the correct intent. Many phonemes have a low importance score or close to 0, and that can be expected because it is logical that a single phoneme does not carry a high amount of information by itself, especially for the phonemes that belong to words that are not associated with a certain intent slot. Others instead carry a lot of importance by themselves:

- **T, ER, N**: These are the phonemes associated with the word *"Turn"*, and even

	T	ER	N	AH	P	DH	AH
action=increase	0.000	0.000	0.001	0.175	0.006	0.003	0.001
object=heat	0.000	0.000	0.000	0.000	0.000	0.000	0.000
location=bedroom	-0.001	-0.001	0.004	0.020	0.004	-0.001	0.738

	B	EH	D	R	UW	M	HH	IY	T
action=increase	0.002	0.000	0.001	0.000	0.002	0.001	0.002	0.116	-0.000
object=heat	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
location=bedroom	0.022	0.968	0.020	-0.001	0.011	0.001	0.002	0.027	0.000

Table 6.2. **Phoneme-level explanations with LOO** for the FSC dataset sample, a higher value and deeper color corresponds to more relevance for the prediction. To fit the page, the table was visually split in two rows, and the vertical lines help distinguish between words. Results are approximated to 3 decimals.

though it carries slight importance for the action *increase* at word level, we can notice that its phonemes carry no importance by themselves for any intent. This could be due to the fact that, from a linguistic point-of-view, they do not distinguish the intent when removed individually, or maybe due to their duration or less distinct pronunciation.

- **AH, P**: The first phoneme of the word *"Up"* carries slight importance, about 20% for the category *increase*. This might mean that the phoneme *AH* is sufficient to identify the intent, or that the *P* sound is not pronounced clearly or too short to carry information on its own (durations are shown in Figure 6.2) . We can see a big difference when comparing the individual results to word level, meaning that their aggregation makes a big difference in the computation of the explanation.
- **DH, AH**: They compose the word *"The"*, which we might assume has no importance for the intents we are recognizing, but we can clearly see that the phoneme *AH* carries a lot of importance for the recognition of the location. We can hypothesize some reasons for this:
  - The alignment is not working correctly, identifying the wrong timestamps for this phoneme, which might be instead belonging to the following phoneme *B*.
  - The classification model uses different clues than what the humans might expect to identify this intent, which goes against the concept of plausibility.
  - Even though the timestamps identified might be correct, the pronunciation of some phonemes, even from different words, can cause the process called *sandhi*, where two phonemes are merged or blended together [84].

  A combination of them could also be the real reason, and to better understand this we could use a different alignment model to see if this process persists.
- **B, EH, D, R, UW, M**: They compose the word *"bedroom"*, and all of the phonemes have very low importance scores aside from *EH*, which carries the most and is able to

uniquely identify the correct location. This could be due to how the model internally works, or because of misalignment issues.

- **HH, IY, T**: The final word *"heat"*, in which *IY* is the only phoneme carrying slight importance, both for the action and the location, but none for the object which we want to identify. This means that removing each of this phoneme individually does not cause any difference in the output prediction by themselves, even though *IY* occupies the majority of the duration of the word, as seen in Figure 6.2.

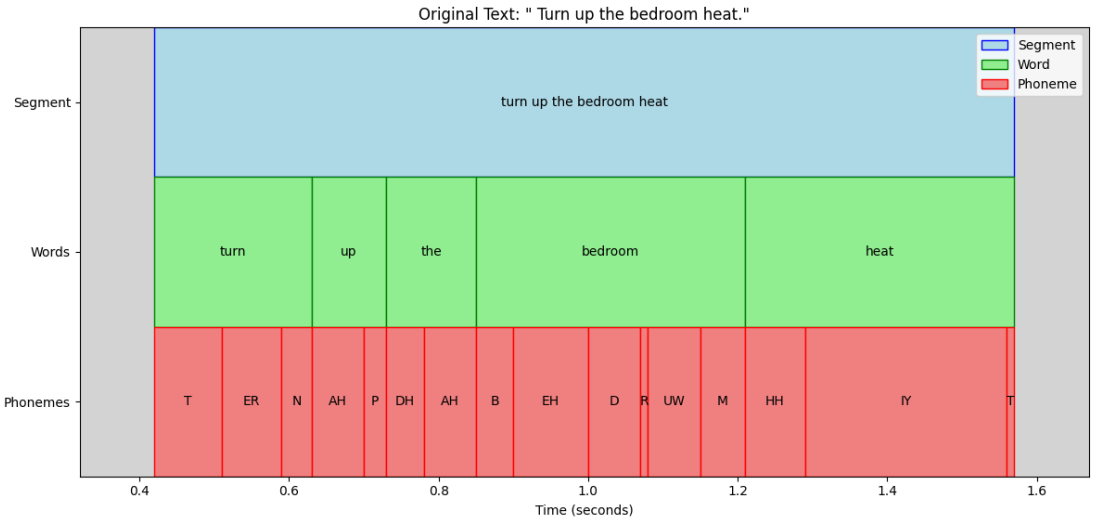

Figure 6.2. The figure shows the timestamp of the sentence, each word and phoneme.

Analyzing the result under a different perspective, we can confirm or deny some of the previous claims. For example, with the representation in Figure 6.3, we can see how the timestamps for each phoneme compare to the waveform of the audio and their importance scores.

By analyzing it, we can make some extra considerations.

- **P, B, D, T** all have a very short duration and the waveform always diminishes, tending to 0 for the last *T* both in amplitude and duration. These phonemes are called *stops* and can be realized without sound release, especially for the last phoneme of a word [85].
- **AH, EH, IY, UH** and all the vowels tend to have more energy thus showing an increase in amplitude [86].
- Some phonemes, like the **R** could barely have a duration due to their pronunciation, making their impact minimal [85].

One important thing to note is that phoneme alignment models work with a **grapheme-to-phoneme approach**. What this means is that they work on a transcription of the

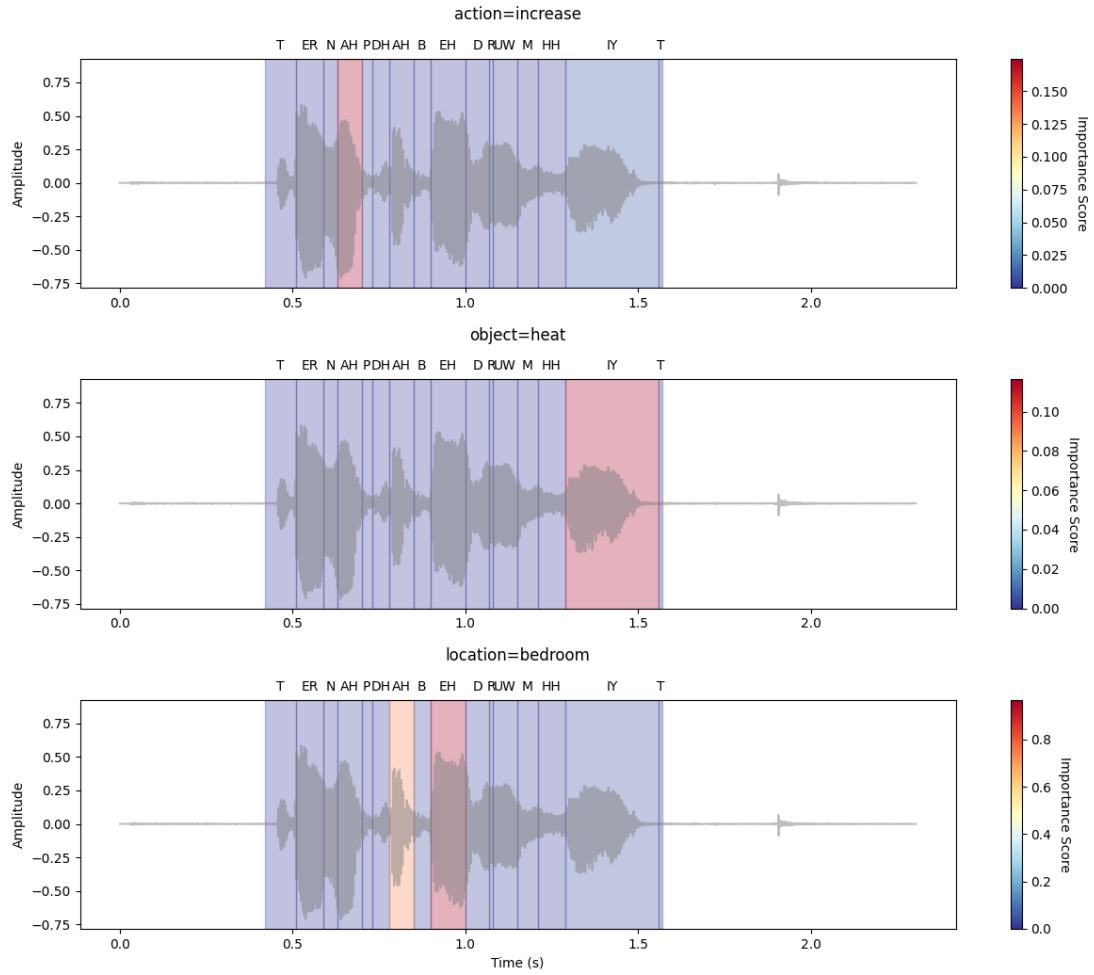

Figure 6.3. Waveform of the FSC audio sample with phoneme alignment and importance scores color-coded.

utterance, then convert the transcription to ARPAbet and then try to align all of them to the audio, even if a certain phoneme was **not actually pronounced**. Methods that try to dynamically create phonemic transcriptions and align the audio have been found to be under performing compared to this approach.

LIME Analyzing the problem with LIME, results in Table 6.3, we can further analyze the problem. First, LIME evaluates importance scores based on local perturbations around a specific prediction, which gives more nuanced results compared to the sharp results of LOO. This reflects especially in phonemes that had 0 as score, and even though they increased slightly they still do not carry enough information to be considered influent. Aside from this, similar observations of the LOO method can be carried. *AH* has the

most importance for the action *increase*, for location *bedroom* both *AH*, from the word *"the"* and *EH*, from *"bedroom"* show the highest scores, and for the object *heat* we have an improvement of 15% circa on the importance of the phoneme *IY* of the word *heat*. This confirms how long vowels seem to carry more information.

	T	ER	N	AH	P	DH	AH
action=increase	0.037	0.015	0.069	0.268	0.054	0.062	0.093
object=heat	0.003	-0.008	0.010	0.001	0.006	-0.008	0.029
location=bedroom	-0.038	-0.029	0.021	0.019	0.027	-0.012	0.401

	B	EH	D	R	UW	M	HH	IY	T
action=increase	0.039	0.013	0.060	0.019	0.061	0.081	0.106	0.076	0.021
object=heat	0.007	0.004	-0.008	-0.003	-0.008	0.003	0.037	0.163	-0.001
location=bedroom	-0.016	0.647	0.074	-0.033	-0.000	0.020	0.053	0.086	-0.024

Table 6.3. **Phoneme-level explanations with LIME** for the FSC dataset sample, a higher value and deeper color corresponds to more relevance for the prediction. To fit the page, the table was visually split in two rows, and the vertical lines help distinguish between words. Results are approximated to 3 decimals.

SHAP Finally, we can conduct the same analysis with the SHAP, results in Table 6.4. This method reinforces what was previously said, providing an interpretation with stable phoneme contribution and possible acoustic blending.

	T	ER	N	AH	P	DH	AH
action=increase	0.017	0.025	0.062	0.168	0.031	0.051	0.056
object=heat	0.003	0.003	-0.012	0.003	0.007	-0.011	0.012
location=bedroom	-0.021	-0.016	-0.008	0.026	0.036	-0.013	0.200

	B	EH	D	R	UW	M	HH	IY	T
action=increase	0.036	0.028	0.056	0.016	0.049	0.035	0.060	0.120	0.005
object=heat	-0.006	0.006	0.002	-0.011	-0.009	-0.003	0.035	0.107	-0.002
location=bedroom	-0.032	0.346	0.074	-0.019	0.024	0.020	0.025	0.096	0.000

Table 6.4. **Phoneme-level explanations with SHAP** for the FSC dataset sample, a higher value and deeper color corresponds to more relevance for the prediction. To fit the page, the table was visually split in two rows, and the vertical lines help distinguish between words. Results are approximated to 3 decimals.

One more thing we can do is **listen to the audios produced** by the removal of the features. This is achieved in the methods by setting the parameter `verbose=True`, allowing us to hear each audio after the perturbation. Even though understanding phoneme timings require lots of training and expertise, an approximate evaluation can be done by an amateur listener. What this shows is that **phoneme alignments are mostly right**, with

the least accuracy in recognizing the *DH* of the word *"the"* and the *B* of *"bedroom"*. Other problematic phonemes are the *P* of *"up"*, *R* of *"bedroom"*, the *HH* and *T* of *"heat"*, but this is mostly because the first two have a very short duration and are barely pronounced, and the last two are not pronounced at all. Conversely, the phoneme *AH* of the word *"the"* is actually **correctly recognized**, showing how the classification model gives high importance to phonemes that we would not associate with certain intent.

Evaluation Metrics Here we utilize the previously introduced metrics in subsection 4.6.5 to evaluate the explanation we have found. The methods output a single value, which represents the average change in the model's prediction probabilities across different thresholds, showing how important or sufficient the high-importance features are for the model's decision. The evaluation can be carried for all the methods we use and based on the results we can understand which one worked the best at identifying the most relevant features. For each method we will evaluate both the comprehensiveness and the sufficiency scores to understand which method worked best for this specific sample.

Before analyzing the results, we must remember that for the comprehensiveness a higher score means that the removed features were important for the model's prediction, whereas a lower score means they didn't affect the prediction much, and the explainer didn't actually find the truly important parts. On the other hand, for the sufficiency evaluator a lower score means that it found the important features. So, for an explanation to be faithful, we would expect a high comprehensiveness (removing important features causes big losses) and a low sufficiency (retaining only the most important parts is enough for the prediction).

For the **LOO explainer**, the comprehensiveness and sufficiency scores are found in Table 6.5.

Evaluator (LOO)	Action = Increase	Object = Heat	Location = Bedroom
Comprehensiveness	0.879	0.765	0.942
Sufficiency	0.624	0.030	0.687

Table 6.5. Evaluation for the LOO explanation for the FSC sample, where a deeper green indicates a good result and orange a result that could be improved.

As we can see the comprehensiveness suggest a highly faithful explanation for all three intents, and we also have a very low sufficiency for `object = heat`. Even though for the other two intents the sufficiency is higher, we can still conclude that the method is able to identify the important features at a good level.

For LIME, the results are in Table 6.6.

They show a similar situation to the results of the LOO method, with very good results for the comprehensiveness and for `object = heat` in case of sufficiency. A higher value of sufficiency might mean that some context might have been missed when retaining only the most important features.

Evaluator (LIME)	Action = Increase	Object = Heat	Location = Bedroom
Comprehensiveness	0.901	0.707	0.997
Sufficiency	0.694	0.002	0.817

Table 6.6. Evaluation for the LOO explanation for the FSC sample, where a deeper green indicates a good result and orange a result that could be improved.

And finally for the **SHAP method**, the scores are as follows in Table 6.7:

Evaluator (SHAP)	Action = Increase	Object = Heat	Location = Bedroom
Comprehensiveness	0.899	0.462	0.923
Sufficiency	0.636	0.003	0.790

Table 6.7. Evaluation for the SHAP explanation for the FSC sample, where a deeper green indicates a good result and orange a result that could be improved.

For the `action = increase` and for `location = bedroom` we have a strong comprehensiveness score and a relatively lower sufficiency, meaning the explanations are quite faithful. On the other hand, `object = heat` presents a very low sufficiency, which is a good sign, but a low comprehensiveness as well, meaning that the model might be able to use other clues to get to the same result, even when removing the most important feature.

When comparing comprehensiveness, LIME and LOO give comparable results while SHAP presents a drop for one of the intents. In terms of sufficiency, LOO performs the best across the three intents. When comparing results it's important to always check these scores as well to understand which method is giving the best explanation.

Parameter Tuning We will now see different configurations of phoneme aggregation and quickly analyze the results with tables. We will start with the following configuration of parameters: `windows_size = 2, respect_word_boundaries = True, sliding = False`, meaning that we aggregate phonemes in group of 2 without allowing repetitions of the same phoneme in different aggregations and avoiding aggregations across words. Aggregations of two phonemes often represent syllables, which can carry more information than the single phoneme.
In Table 6.8 we can see that for the LOO methodology, *DHAH* and *BEH* uniquely identify the location, *HHIY* carries the prediction for the object and, as expected, *AHP* has the most importance for the action. It's important to remember that these scores are not aggregated from the single phonemes scores, but instead treated as a feature in itself.

And similarly, in Table 6.9, the results for the same configuration for SHAP. We can see again how the results are more spread out and have a lower magnitude.

If we activate the `sliding` parameter, we achieve the results seen in Table 6.10 with LOO. For each word, the phonemes are aggregated two by two, creating features such

	TER	N	AHP	DHAH
action=increase	0.002	0.001	0.271	0.001
object=heat	0.000	0.000	0.000	0.000
location=bedroom	-0.001	0.004	0.024	0.626

	BEH	DR	UWM	HHIY	T
action=increase	0.001	0.002	0.001	0.011	0.000
object=heat	0.000	0.000	0.000	0.666	0.000
location=bedroom	0.954	0.020	0.024	0.139	0.002

Table 6.8. Phoneme-level explanations with LOO for the FSC dataset sample, where higher values and deeper colors indicate more relevance. The table is split in two to fit the page. Parameters: `windows_size = 2, respect_word_boundaries = True, sliding = False`.

	TER	N	AHP	DHAH
action=increase	0.022	0.053	0.263	0.091
object=heat	0.014	-0.032	-0.003	-0.017
location=bedroom	-0.021	0.003	0.035	0.185

	BEH	DR	UWM	HHIY	T
action=increase	0.021	0.066	0.046	0.123	0.004
object=heat	-0.002	-0.023	-0.011	0.353	-0.000
location=bedroom	0.295	0.087	0.018	0.183	-0.001

Table 6.9. Phoneme-level explanations with SHAP for the FSC dataset sample, where higher values and deeper colors indicate more relevance. The table is split in two for readability. Parameters: `windows_size = 2, respect_word_boundaries = True, sliding = False`.

as *TER*, from *T* and *ER*, and *ERN*, from *ER* and *N*, showing how *ER* repeats acorss different features. Even though this specific case does not show much difference, for a different sample analyzing the audio with different combination of parameters could show varying scores across aggregations.

6.2 Multi-language analysis

In this section, we will analyze the same utterance spoken in different languages, to analyze how the differences in languages and how the models perform under different conditions. To do this, we will use the ITALIC dataset for Italian audios and Speech Massive for German and French audios. We compare the same transcription, in this case *"What is the weather for this week?"* in English, which corresponds to `id = 5038` in the train split for ITALIC, while for the rest of the dataset it appears in the corresponding test split. The audio has been randomly chosen and for each language an assessment of the audio quality was done by listening to the audio, checking if the pronunciation sounds

	TER	ERN	AHP	DHAH
action=increase	0.002	0.003	0.271	0.001
object=heat	0.000	0.000	0.000	0.000
location=bedroom	-0.001	0.002	0.024	0.626

	BEH	EHD	DR	RUW	UWM	HHIY	IYT
action=increase	0.001	0.000	0.002	0.002	0.001	0.011	0.007
object=heat	0.000	0.001	-0.000	-0.000	0.000	0.666	0.115
location=bedroom	0.954	0.996	0.020	0.018	0.024	0.139	0.029

Table 6.10. Phoneme-level explanations with LOO for the FSC dataset sample, where higher values and deeper colors indicate more relevance. The table is split in two for readability. Parameters: `windows_size = 2, respect_word_boundaries = True, sliding = True`.

clear.
Furthermore, for this analysis we use a different phoneme alignment model, since the one previously used only supports English utterances, called *"facebook/wav2vec2-lv-60-espeak-cv-ft"* from Meta [74]. Unlike Charsiu's model, this one supports multiple languages while possibly losing accuracy since we do not have a comparison of the timestamps with the benchmark dataset TIMIT [87]. On top of this, this model returns phonetic timestamps, thus we will use IPA alphabet for the analysis since IPA is an internationally-accepted phonetic dictionary unlike ARPABET which is built on top of American-English pronunciation. Since a qualitative evaluation of the phoneme timestamps is not possible for all the mentioned languages, we will only analyze the quantitative results provided, and carry a qualitative evaluation only for the Italian sample.

Italian Starting off with the Italian instance, *"Che tempo fa questa settimana?"* = *"What is the weather for this week?"*, the following are the results for the alignment (Figure 6.4):

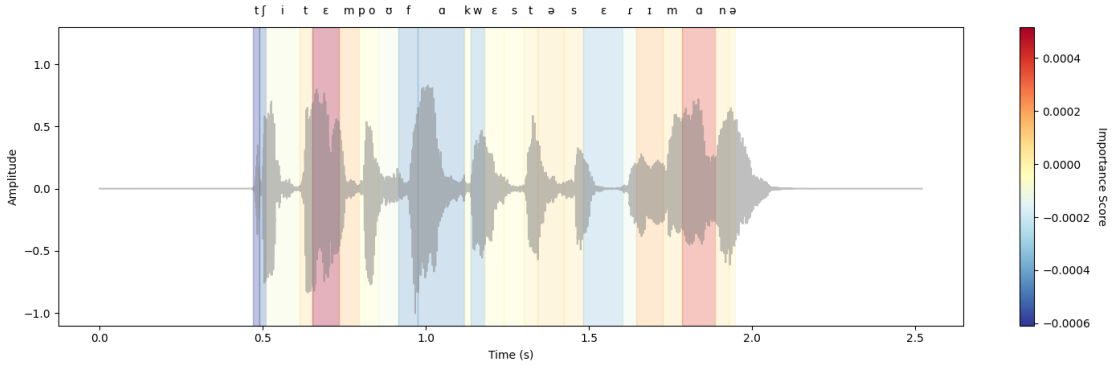

Figure 6.4. Phoneme-level explanations with LOO for the ITALICS dataset sample *"Che tempo fa questa settimana?"*, where higher values indicate more relevance.

Immediately we can notice how the representation of the phoneme has changed, using now the IPA representation to be conformed with the output of the multi-language model. Another important thing we can notice is that the alignment is probably not correct, because the final part of the audio (in Figure 6.4 represented by the gray amplitude) is not covered by any phoneme, which is unlikely. To confirm this, we can manually assess the audio, even though the high quantity of phonemes and their short duration makes it hard to evaluate their correctness. As expected, some of the phonemes are correctly aligned, but others are not (such as the phonemes in the last word of the sentence). This could be due to poor training of the model used for alignment, since the audio is correctly pronounced and has high and clear quality. On average, we can also see that the phonemes have a very short time aside from some of the vowels. This reflects in the results, which highlights a situation where all the importance scores are very close to 0, even when aggregating multiple phonemes. This is a weird result and could be due to the robustness of the model to small changes in the input.
With a different audio, `id=71` saying *"Mi piacciono le canzoni di Bocelli" = "I like Bocelli's songs"*, we get the output in Figure 6.5.

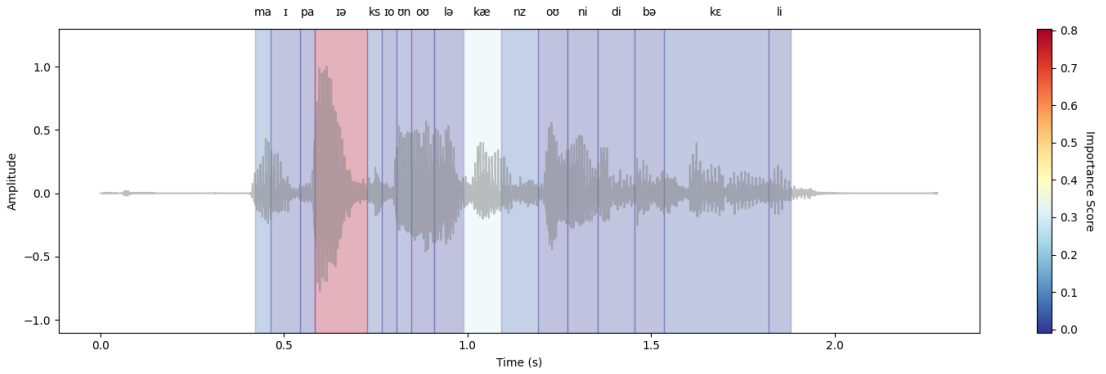

Figure 6.5. Phoneme-level explanations with LOO for the ITALICS dataset sample *"Mi piacciono le canzoni di Bocelli"*, where higher values indicate more relevance. Parameters: `windows_size = 2, respect_word_boundaries = True, sliding = False`.

For this example we use an aggregation window of size 2, to capture more information, otherwise we would find results close to 0 once again. Here we can see that the alignment already seem to perform better, and two groups of phonemes carry most of the importance in a much greater magnitude than the previous example. Particularly, the second group of the word *"piacciono"* and the first of *"canzoni"* have high importance to recognize `intent=music_likeness`. Not by chance, they correspond to the words *"like"* and *"songs"*. A qualitative assessment confirms that these phoneme groups are correctly identified, and once again vowels carry the highest importance in the sentence.

German Similarly for the German sample, if we analyze the same initial sample of the ITALICS dataset, we get similar bad results for the sentence *"Wie wird diese Woche das Wetter?" = "What is the weather for this week?"* (Figure 6.6):

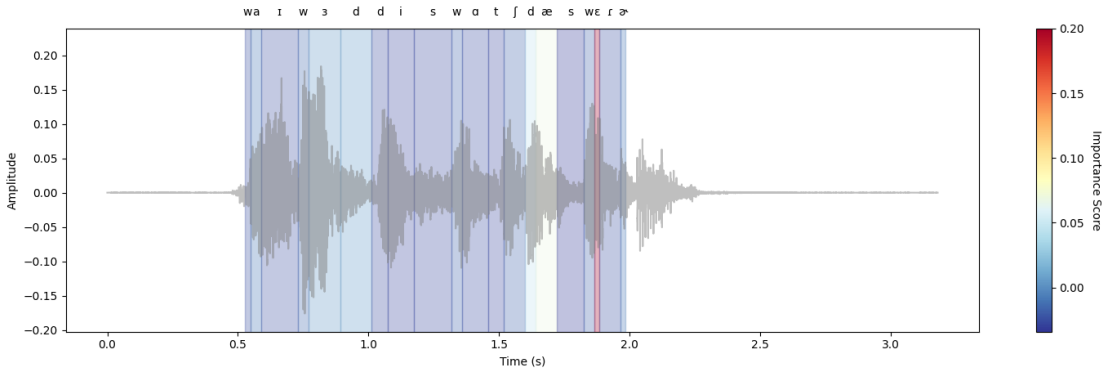

Figure 6.6. Phoneme-level explanations with LOO for the Speech MASSIVE dataset sample *"Wie wird diese Woche das Wetter?"*, where higher values indicate more relevance.

Similarly to before, it fails to capture correctly the end of the sentence, but this time we cannot carry a qualitative assessment due to language limitations. The color-coded results also show that the highest magnitude is for a phoneme inside the word *"Wetter"* = *"Weather"* which has very short duration, and is probably not correctly recognized, but this might also suggest which part of the utterances carry more information.
The second sample, `id = 71`, contains *"Ich mag Lindenberg-Lieder"*, which translates to *"I like Lindenberg's songs"* (Figure 6.7). This sample also shows low accuracy in the alignment, making the analysis hard to carry.

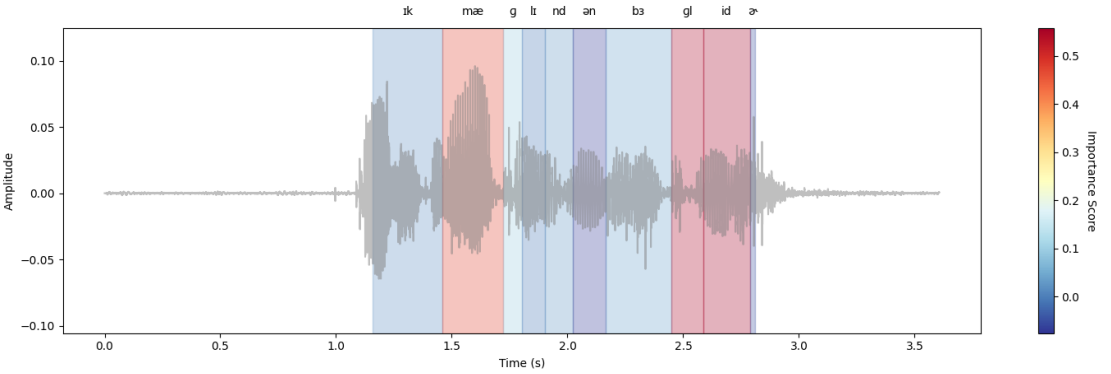

Figure 6.7. Phoneme-level explanations with LOO for the Speech MASSIVE dataset sample *"Ich mag Lindenberg-Lieder"*, where higher values indicate more relevance. Parameters: `windows_size = 2, respect_word_boundaries = True, sliding = False`.

French Finally, the first French sample we analyze, `id = 5038` with transcription *"Quelle est la météo pour cette semaine?"* = *"What is the weather for this week?"*, shows apparently better alignment results (Figure 6.8).

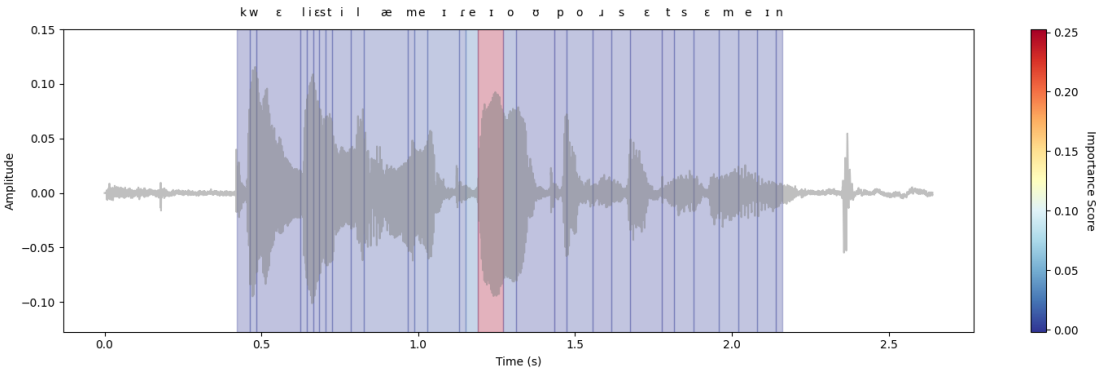

Figure 6.8. Phoneme-level explanations with LOO for the ITALICS dataset sample *"Quelle est la météo pour cette semaine?"*, where higher values indicate more relevance.

Analyzing the audio, we can hear that some of the final phonemes are slightly misaligned again, while the spike that we can see at the end of the audio is caused by a low bump sound. We can also see that there is a single phoneme with higher importance than the others, the second "ɪ" of the word *"météo"* (= *"weather"*. Listening to that sample, we can hear that it corresponds to part of the final sound of that word, so we can assume it is correctly identified and rightfully carries importance in identifying the intent. An ear with a better French understanding could confirm this hypothesis.
For the second sample, `id = 71` with transcription *"J'aime les chansons de Jacques Brel"* which translates to *"I like Jacques Brel songs"*, we see the first word (*"J'"* = "I") having a high importance (Figure 6.9:

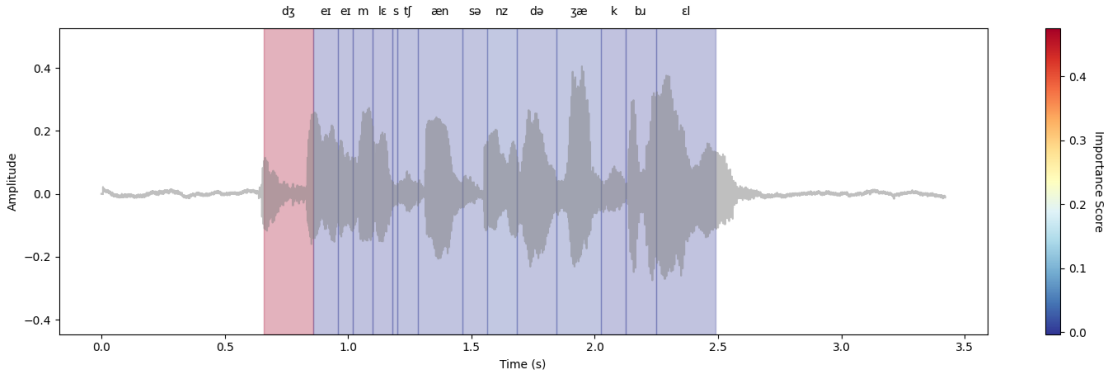

Figure 6.9. Phoneme-level explanations with LOO for the Speech MASSIVE dataset sample *"J'aime les chansons de Jacques Brel"*, where higher values indicate more relevance. Parameters: `windows_size = 2, respect_word_boundaries = True, sliding = False`.

Listening to the modified audios we can hear that the first group (which only identifies a single letter *"J"*) leaks into the second word, cutting away part of the word *"aime"* (=

"like"). This is due to poor alignment, but it shows how heavily the model relies on a small part of the input, only 0.2 seconds long.

In addition to what has been reported in this thesis, there are still thousands of samples to be tested. For example, for other samples the classification model failed to correctly identify the intent, and for other the alignment model performed accurately. Still, it is evident that the different model introduced a heavy drop in performance when we look at the samples provided. This underlines how even one of the best models available has still great improvement potential, and how there is still room for further research into the role of phonemes in ASR.

Chapter 7

Conclusion

This thesis addressed the challenges of improving the interpretability for speech recognition models, specifically the objective was moving from word-level explanations to phoneme-level. This sets a foundation for further and more detailed analyses of the black-box models that are used for automatic speech recognition, shedding light on a problem that was yet to be explored at this granularity.
Specifically, this approach introduced customizable phoneme explanations, with aggregations and sliding windows techniques that enables the exploration of the models at different granularities, from phonemes to syllables and morphemes. This transition allowed us to have a new view of how the chosen model makes decisions, leveraging new audio alignment techniques that have high accuracy in this task. Through the adaptation of different methodologies (LOO, LIME, and SHAP), we are able to obtain the results we have previously analyzed and understand which phonemes carry the most importance. What they showed is that phonemes play an important role in determining predictions, even though at first we might think they are too short to make a difference in the predictions. This is confirmed through the analyses of the sufficiency and comprehensiveness results.
When switching model for the multi-language analysis we can see a drastic drop in accuracy for the alignment of phonemes. This is a hard task and the exploration of better models might be explored. This includes language-specific alignment models or other alignment tools. For the former, currently there are not many models that can output timestamps with this granularity for specific languages, and the literature has only worked out solutions for English and Chinese languages. Other alignment tools, like Montreal Forced Aligner (MFA) [88], were incompatible with the current implementation of certain methods, thus making them unusable for this project.

These results directly enable transparency on the models, increasing user trust if needed in sensitive applications where understanding a prediction might be crucial. Moreover, it allows the ability to identify, and if needed correct, some patterns that drive a certain prediction. For example, we see that the phoneme AH of the word *"the"* is highly influential for the result of *location* in Table 6.2. This goes against what a human might expect, but

further investigation could help understand the reason. This could also help in identifying and correcting biases, for example for certain accents, and through the paralinguistic perturbations at phoneme level we might be able to understand where a model fails.

Future improvements might include how to implement a multi-lingual phoneme recognition model or a model that dynamically create phoneme representation instead of using a grapheme-to-phoneme approach.

Bibliography

[1] Alejandro Barredo Arrieta, Natalia Díaz-Rodríguez, Javier Del Ser, Adrien Bennetot, Siham Tabik, Alberto Barbado, Salvador García, Sergio Gil-López, Daniel Molina, Richard Benjamins, Raja Chatila, and Francisco Herrera. Explainable artificial intelligence (xai): Concepts, taxonomies, opportunities and challenges toward responsible ai, 2019. URL https://arxiv.org/abs/1910.10045.

[2] Alkis Koudounas, Flavio Giobergia, Eliana Pastor, and Elena Baralis. A contrastive learning approach to mitigate bias in speech models. In *Interspeech 2024*, interspeech 2024, page 827–831. ISCA, September 2024. doi: 10.21437/interspeech.2024-1219. URL http://dx.doi.org/10.21437/Interspeech.2024-1219.

[3] Neema Mishra, Urmila Shrawankar, and V M Thakare. Automatic speech recognition using template model for man-machine interface, 2013. URL https://arxiv.org/abs/1305.2959.

[4] Nagajyothi Dimmita and P. Siddaiah. Speech recognition using convolutional neural networks. *International Journal of Engineering and Technology(UAE)*, 7:133–137, 09 2018. doi: 10.14419/ijet.v7i4.6.20449.

[5] Anmol Gulati, James Qin, Chung-Cheng Chiu, Niki Parmar, Yu Zhang, Jiahui Yu, Wei Han, Shibo Wang, Zhengdong Zhang, Yonghui Wu, and Ruoming Pang. Conformer: Convolution-augmented transformer for speech recognition, 2020. URL https://arxiv.org/abs/2005.08100.

[6] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2023. URL https://arxiv.org/abs/1706.03762.

[7] Alexei Baevski, Henry Zhou, Abdelrahman Mohamed, and Michael Auli. wav2vec 2.0: A framework for self-supervised learning of speech representations, 2020. URL https://arxiv.org/abs/2006.11477.

[8] Alec Radford, Jong Wook Kim, Tao Xu, Greg Brockman, Christine McLeavey, and Ilya Sutskever. Robust speech recognition via large-scale weak supervision, 2022. URL https://arxiv.org/abs/2212.04356.

[9] Siyuan Feng, Bence Mark Halpern, Olya Kudina, and Odette Scharenborg. Towards inclusive automatic speech recognition. *Computer Speech Language*, 84:101567, 2024. ISSN 0885-2308. doi: https://doi.org/10.1016/j.csl.2023.101567. URL https://www.sciencedirect.com/science/article/pii/S0885230823000864.

[10] Siyuan Feng, Olya Kudina, Bence Mark Halpern, and Odette Scharenborg. Quantifying bias in automatic speech recognition, 2021. URL https://arxiv.org/abs/2103.15122.

[11] Alkis Koudounas, Eliana Pastor, Giuseppe Attanasio, Vittorio Mazzia, Manuel Giollo, Thomas Gueudre, Luca Cagliero, Luca de Alfaro, Elena Baralis, and Daniele Amberti. Exploring subgroup performance in end-to-end speech models. In *ICASSP 2023 - 2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1–5, 2023. doi: 10.1109/ICASSP49357.2023.10095284.

[12] Allison Koenecke, Andrew Nam, Emily Lake, Joe Nudell, Minnie Quartey, Zion Mengesha, Connor Toups, John R. Rickford, Dan Jurafsky, and Sharad Goel. Racial disparities in automated speech recognition. *Proceedings of the National Academy of Sciences*, 117(14):7684–7689, 2020. doi: 10.1073/pnas.1915768117. URL https://www.pnas.org/doi/abs/10.1073/pnas.1915768117.

[13] PRANAV DHERAM, Murugesan Ramakrishnan, Anirudh Raju, I-Fan Chen, Brian King, Katherine Powell, Melissa Saboowala, Karan Shetty, and Andreas Stolcke. Toward fairness in speech recognition: Discovery and mitigation of performance disparities. In *Interspeech 2022*, interspeech 2022, page 1268–1272. ISCA, September 2022. doi: 10.21437/interspeech.2022-10816. URL http://dx.doi.org/10.21437/Interspeech.2022-10816.

[14] Xingyu Chen, Zhengxiong Li, Srirangaraj Setlur, and Wenyao Xu. Exploring racial and gender disparities in voice biometrics. *Scientific Reports*, 12:3723, 03 2022. doi: 10.1038/s41598-022-06673-y.

[15] Rita Frieske and Bertram E. Shi. Hallucinations in neural automatic speech recognition: Identifying errors and hallucinatory models, 2024. URL https://arxiv.org/abs/2401.01572.

[16] Hamza Kheddar, Mustapha Hemis, and Yassine Himeur. Automatic speech recognition using advanced deep learning approaches: A survey. *Information Fusion*, 109: 102422, September 2024. ISSN 1566-2535. doi: 10.1016/j.inffus.2024.102422. URL http://dx.doi.org/10.1016/j.inffus.2024.102422.

[17] Scott M Lundberg and Su-In Lee. A unified approach to interpreting model predictions. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/8a20a8621978632d76c43dfd28b67767-Paper.pdf.

[18] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. "why should i trust you?": Explaining the predictions of any classifier, 2016. URL https://arxiv.org/abs/1602.04938.

[19] Xiaoliang Wu, Peter Bell, and Ajitha Rajan. Explanations for automatic speech recognition, 2023. URL https://arxiv.org/abs/2302.14062.

[20] Hana Chockler, Daniel Kroening, and Youcheng Sun. Compositional explanations for image classifiers, 03 2021.

[21] Leila Arras, José Arjona-Medina, Michael Widrich, Grégoire Montavon, Michael Gillhofer, Klaus-Robert Müller, Sepp Hochreiter, and Wojciech Samek. Explaining and interpreting lstms. In *Explainable AI: Interpreting, Explaining and Visualizing Deep Learning*, pages 211–238. Springer, 2019.

[22] Sebastian Lapuschkin, Alexander Binder, Grégoire Montavon, Frederick Klauschen, Klaus-Robert Müller, and Wojciech Samek. On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. *PLoS ONE*, 10:e0130140, 07 2015. doi: 10.1371/journal.pone.0130140.

[23] Sören Becker, Marcel Ackermann, Sebastian Lapuschkin, Klaus-Robert Müller, and Wojciech Samek. Interpreting and explaining deep neural networks for classification of audio signals, 07 2018.

[24] Annika Frommholz, Fabian Seipel, Sebastian Lapuschkin, Wojciech Samek, and Johanna Vielhaben. Xai-based comparison of input representations for audio event classification, 2023. URL https://arxiv.org/abs/2304.14019.

[25] Marco Colussi and Stavros Ntalampiras. Interpreting deep urban sound classification using layer-wise relevance propagation, 2021. URL https://arxiv.org/abs/2111.10235.

[26] Xiaoliang Wu, Peter Bell, and Ajitha Rajan. Explanations for automatic speech recognition. In *ICASSP 2023 - 2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1–5, 2023. doi: 10.1109/ICASSP49357.2023.10094635.

[27] Saumitra Mishra, Bob L. Sturm, and Simon Dixon. Local interpretable model-agnostic explanations for music content analysis. In Sally Jo Cunningham, Zhiyao Duan, Xiao Hu, and Douglas Turnbull, editors, *Proceedings of the 18th International Society for Music Information Retrieval Conference, ISMIR 2017, Suzhou, China, October 23-27, 2017*, pages 537–543, 2017. URL https://ismir2017.smcnus.org/wp-content/uploads/2017/10/216_Paper.pdf.

[28] Eliana Pastor, Alkis Koudounas, Giuseppe Attanasio, Dirk Hovy, and Elena Baralis. Explaining speech classification models via word-level audio segments and paralinguistic features. In Yvette Graham and Matthew Purver, editors, *Proceedings of the 18th Conference of the European Chapter of the Association for Computational*

Linguistics (Volume 1: Long Papers), pages 2221–2238, St. Julian's, Malta, March 2024. Association for Computational Linguistics. URL https://aclanthology.org/2024.eacl-long.136/.

[29] Ian Covert, Scott Lundberg, and Su-In Lee. Explaining by removing: A unified framework for model explanation, 2022. URL https://arxiv.org/abs/2011.14878.

[30] Xiaoliang Wu, Peter Bell, and Ajitha Rajan. Can we trust explainable ai methods on asr? an evaluation on phoneme recognition, 2023. URL https://arxiv.org/abs/2305.18011.

[31] PRANAV DHERAM, Murugesan Ramakrishnan, Anirudh Raju, I-Fan Chen, Brian King, Katherine Powell, Melissa Saboowala, Karan Shetty, and Andreas Stolcke. Toward fairness in speech recognition: Discovery and mitigation of performance disparities. In *Interspeech 2022*, pages 1268–1272, 2022. doi: 10.21437/Interspeech.2022-10816.

[32] Alkis Koudounas, Eliana Pastor, Giuseppe Attanasio, Vittorio Mazzia, Manuel Giollo, Thomas Gueudre, Luca Cagliero, Luca de Alfaro, Elena Baralis, and Daniele Amberti. Exploring subgroup performance in end-to-end speech models. In *ICASSP 2023 - 2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1–5, 2023. doi: 10.1109/ICASSP49357.2023.10095284.

[33] Yuanyuan Zhang, Aaricia Herygers, Tanvina Patel, Zhengjun Yue, and Odette Scharenborg. Exploring data augmentation in bias mitigation against non-native-accented speech, 2023. URL https://arxiv.org/abs/2312.15499.

[34] Yuanyuan Zhang, Yixuan Zhang, Bence Halpern, Tanvina Patel, and Odette Scharenborg. Mitigating bias against non-native accents. pages 3168–3172, 09 2022. doi: 10.21437/Interspeech.2022-836.

[35] Irina-Elena Veliche and Pascale Fung. Improving fairness and robustness in end-to-end speech recognition through unsupervised clustering, 2023. URL https://arxiv.org/abs/2306.06083.

[36] Alkis Koudounas, Eliana Pastor, Giuseppe Attanasio, Luca de Alfaro, and Elena Baralis. Prioritizing data acquisition for end-to-end speech model improvement. In *ICASSP 2024 - 2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 7000–7004, 2024. doi: 10.1109/ICASSP48485.2024.10446326.

[37] John S. Garofolo, Lori F. Lamel, William M. Fisher, Jonathan G. Fiscus, David S. Pallett, Nancy L. Dahlgren, and Victor Zue. Timit acoustic-phonetic continuous speech corpus, 1993. URL https://catalog.ldc.upenn.edu/LDC93S1. LDC Catalog No. LDC93S1.

[38] Jian Zhu, Cong Zhang, and David Jurgens. Phone-to-audio alignment without text: A semi-supervised approach. *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2022.

[39] Michael McAuliffe, Michaela Socolof, Sarah Mihuc, Michael Wagner, and Morgan Sonderegger. Montreal Forced Aligner: Trainable Text-Speech Alignment Using Kaldi. In *Proc. Interspeech 2017*, pages 498–502, 2017. doi: 10.21437/Interspeech.2017-1386.

[40] European Parliament and Council of the European Union. Regulation (EU) 2016/679 of the European Parliament and of the Council. URL https://data.europa.eu/eli/reg/2016/679/oj.

[41] Alexander Heemann, Desislava Velinova, and Mario Gastegger. Textklassification - decision trees random forests, 07 2017.

[42] Michael Hind, Dennis Wei, Murray Campbell, Noel C. F. Codella, Amit Dhurandhar, Aleksandra Mojsilović, Karthikeyan Natesan Ramamurthy, and Kush R. Varshney. Ted: Teaching ai to explain its decisions, 2019. URL https://arxiv.org/abs/1811.04896.

[43] Sarah Wiegreffe and Ana Marasović. Teach me to explain: A review of datasets for explainable natural language processing, 2021. URL https://arxiv.org/abs/2102.12060.

[44] Alejandro Barredo Arrieta, Natalia Díaz-Rodríguez, Javier Del Ser, Adrien Bennetot, Siham Tabik, Alberto Barbado, Salvador Garcia, Sergio Gil-Lopez, Daniel Molina, Richard Benjamins, Raja Chatila, and Francisco Herrera. Explainable artificial intelligence (xai): Concepts, taxonomies, opportunities and challenges toward responsible ai. *Information Fusion*, 58:82–115, 2020. ISSN 1566-2535. doi: https://doi.org/10.1016/j.inffus.2019.12.012. URL https://www.sciencedirect.com/science/article/pii/S1566253519308103.

[45] Chirag Agarwal, Sree Harsha Tanneru, and Himabindu Lakkaraju. Faithfulness vs. plausibility: On the (un)reliability of explanations from large language models, 2024. URL https://arxiv.org/abs/2402.04614.

[46] Vanya Van Belle, Ben Van Calster, Sabine Huffel, Johan Suykens, and P.j.g Lisboa. Explaining support vector machines: A color based nomogram. *PLOS ONE*, 11: e0164568, 10 2016. doi: 10.1371/journal.pone.0164568.

[47] Ramprasaath R. Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. *International Journal of Computer Vision*, 128(2):336–359, October 2019. ISSN 1573-1405. doi: 10.1007/s11263-019-01228-7. URL http://dx.doi.org/10.1007/s11263-019-01228-7.

[48] Adrien Bibal, Rémi Cardon, David Alfter, Rodrigo Wilkens, Xiaoou Wang, Thomas François, and Patrick Watrin. Is attention explanation? an introduction to the

debate. In Smaranda Muresan, Preslav Nakov, and Aline Villavicencio, editors, *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3889–3900, Dublin, Ireland, May 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.acl-long.269. URL https://aclanthology.org/2022.acl-long.269/.

[49] Sarthak Jain and Byron C. Wallace. Attention is not explanation, 2019. URL https://arxiv.org/abs/1902.10186.

[50] Sarah Wiegreffe and Yuval Pinter. Attention is not not explanation, 2019. URL https://arxiv.org/abs/1908.04626.

[51] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, page 785–794. ACM, August 2016. doi: 10.1145/2939672.2939785. URL http://dx.doi.org/10.1145/2939672.2939785.

[52] Rex Ying, Dylan Bourgeois, Jiaxuan You, Marinka Zitnik, and Jure Leskovec. Gnnexplainer: Generating explanations for graph neural networks, 2019. URL https://arxiv.org/abs/1903.03894.

[53] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. Model-agnostic interpretability of machine learning, 2016. URL https://arxiv.org/abs/1606.05386.

[54] Leo Breiman. Random forests. 45(1):5–32. doi: 10.1023/A:1010933404324.

[55] Aaron Fisher, Cynthia Rudin, and Francesca Dominici. All models are wrong, but many are useful: Learning a variable's importance by studying an entire class of prediction models simultaneously, 2019. URL https://arxiv.org/abs/1801.01489.

[56] Brandon M. Greenwell, Bradley C. Boehmke, and Andrew J. McCarthy. A simple and effective model-based variable importance measure, 2018. URL https://arxiv.org/abs/1805.04755.

[57] Alex Goldstein, Adam Kapelner, Justin Bleich, and Emil Pitkin. Peeking inside the black box: Visualizing statistical learning with plots of individual conditional expectation, 2014. URL https://arxiv.org/abs/1309.6392.

[58] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps, 2014. URL https://arxiv.org/abs/1312.6034.

[59] Daniel Smilkov, Nikhil Thorat, Been Kim, Fernanda Viégas, and Martin Wattenberg. Smoothgrad: removing noise by adding noise, 2017. URL https://arxiv.org/abs/1706.03825.

[60] Benjamin Letham, Cynthia Rudin, Tyler H. McCormick, and David Madigan. Interpretable classifiers using rules and bayesian analysis: Building a better stroke prediction model. *The Annals of Applied Statistics*, 9(3), September 2015. ISSN 1932-6157. doi: 10.1214/15-aoas848. URL http://dx.doi.org/10.1214/15-AOAS848.

[61] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. Anchors: High-precision model-agnostic explanations. *Proceedings of the AAAI Conference on Artificial Intelligence*, 32(1), Apr. 2018. doi: 10.1609/aaai.v32i1.11491. URL https://ojs.aaai.org/index.php/AAAI/article/view/11491.

[62] Sandra Wachter, Brent Mittelstadt, and Chris Russell. Counterfactual explanations without opening the black box: Automated decisions and the gdpr, 2018. URL https://arxiv.org/abs/1711.00399.

[63] Been Kim, Rajiv Khanna, and Oluwasanmi O Koyejo. Examples are not enough, learn to criticize! criticism for interpretability. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016. URL https://proceedings.neurips.cc/paper_files/paper/2016/file/5680522b8e2bb01943234bce7bf84534-Paper.pdf.

[64] Marko Robnik-Šikonja and Igor Kononenko. Explaining classifications for individual instances. *IEEE Transactions on Knowledge and Data Engineering*, 20(5):589–600, 2008. doi: 10.1109/TKDE.2007.190734.

[65] David Alvarez-Melis and Tommi S. Jaakkola. On the robustness of interpretability methods, 2018. URL https://arxiv.org/abs/1806.08049.

[66] Christopher Burger, Lingwei Chen, and Thai Le. Are your explanations reliable? investigating the stability of lime in explaining text classifiers by marrying xai and adversarial attack, 2023. URL https://arxiv.org/abs/2305.12351.

[67] Dylan Slack, Sophie Hilgard, Emily Jia, Sameer Singh, and Himabindu Lakkaraju. Fooling lime and shap: Adversarial attacks on post hoc explanation methods, 2020. URL https://arxiv.org/abs/1911.02508.

[68] Lloyd S Shapley. A value for n-person games. In Harold W. Kuhn and Albert W. Tucker, editors, *Contributions to the Theory of Games II*, pages 307–317. Princeton University Press, Princeton, 1953.

[69] Erik Štrumbelj and Igor Kononenko. Explaining prediction models and individual predictions with feature contributions. *Knowledge and Information Systems*, 41:647–665, 12 2013. doi: 10.1007/s10115-013-0679-x.

[70] Lars Henry Berge Olsen and Martin Jullum. Improving the sampling strategy in kernelshap, 2024. URL https://arxiv.org/abs/2410.04883.

[71] Neil Jethani, Mukund Sudarshan, Ian Covert, Su-In Lee, and Rajesh Ranganath. Fastshap: Real-time shapley value estimation, 2022. URL https://arxiv.org/abs/2107.07436.

[72] James Robert, Marc Webbie, et al. Pydub, 2018. URL http://pydub.com/.

[73] Max Bain, Jaesung Huh, Tengda Han, and Andrew Zisserman. Whisperx: Time-accurate speech transcription of long-form audio. *INTERSPEECH 2023*, 2023.

[74] Facebook AI. wav2vec2-lv-60-espeak-cv-ft. https://huggingface.co/facebook/wav2vec2-lv-60-espeak-cv-ft, 2021. Accessed: 2025-03-04.

[75] Iver Jordal et al. audiomentations: A python library for audio data augmentation. URL https://github.com/iver56/audiomentations.

[76] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online, October 2020. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/2020.emnlp-demos.6.

[77] Loren Lugosch, Mirco Ravanelli, Patrick Ignoto, Vikrant Singh Tomar, and Yoshua Bengio. Speech model pre-training for end-to-end spoken language understanding, 2019. URL https://arxiv.org/abs/1904.03670.

[78] Shu wen Yang, Po-Han Chi, Yung-Sung Chuang, Cheng-I Jeff Lai, Kushal Lakhotia, Yist Y. Lin, Andy T. Liu, Jiatong Shi, Xuankai Chang, Guan-Ting Lin, Tzu-Hsien Huang, Wei-Cheng Tseng, Ko tik Lee, Da-Rong Liu, Zili Huang, Shuyan Dong, Shang-Wen Li, Shinji Watanabe, Abdelrahman Mohamed, and Hung yi Lee. Superb: Speech processing universal performance benchmark, 2021. URL https://arxiv.org/abs/2105.01051.

[79] Alkis Koudounas, Moreno La Quatra, Lorenzo Vaiani, Luca Colomba, Giuseppe Attanasio, Eliana Pastor, Luca Cagliero, and Elena Baralis. Italic: An italian intent classification dataset. In *INTERSPEECH 2023*, page 2153–2157. ISCA, August 2023. doi: 10.21437/interspeech.2023-1980. URL http://dx.doi.org/10.21437/Interspeech.2023-1980.

[80] Arun Babu, Changhan Wang, Andros Tjandra, Kushal Lakhotia, Qiantong Xu, Naman Goyal, Kritika Singh, Patrick von Platen, Yatharth Saraf, Juan Pino, Alexei Baevski, Alexis Conneau, and Michael Auli. Xls-r: Self-supervised cross-lingual speech representation learning at scale. In *Interspeech 2022*, pages 2278–2282, 2022. doi: 10.21437/Interspeech.2022-143.

[81] Carlos Busso, Murtaza Bulut, Chi-Chun Lee, Ebrahim (Abe) Kazemzadeh, Emily Mower Provost, Samuel Kim, Jeannette N. Chang, Sungbok Lee, and Shrikanth S. Narayanan. Iemocap: Interactive emotional dyadic motion capture database. *Language Resources and Evaluation*, 42:335–359, 2008. doi: 10.1007/s10579-008-9076-6.

[82] International Phonetic Association. International phonetic association, 2025. URL https://www.internationalphoneticassociation.org/. Accessed: 2025-03-10.

[83] June E. Shoup. Phonological aspects of speech recognition. In Wayne A. Lea, editor, *Trends in Speech Recognition*, pages 125–138. Prentice-Hall, Englewood Cliffs, NJ, 1980.

[84] Sanford A. Schane. Sandhi. In Keith Brown, editor, *Encyclopedia of Language and Linguistics*, pages 359–362. Elsevier, Oxford, 2nd edition, 2006.

[85] John Laver. *Principles of Phonetics*. Cambridge Textbooks in Linguistics. Cambridge University Press, 1994.

[86] P. Ladefoged and K. Johnson. *A Course in Phonetics*. Cengage Learning, 2010. ISBN 9781428231269. URL https://books.google.it/books?id=FjLc1XtqJUUC.

[87] J. Garofolo, Lori Lamel, W. Fisher, Jonathan Fiscus, D. Pallett, N. Dahlgren, and V. Zue. Timit acoustic-phonetic continuous speech corpus. *Linguistic Data Consortium*, 11 1992.

[88] Michael McAuliffe, Michaela Socolof, Sarah Mihuc, Michael Wagner, and Morgan Sonderegger. Montreal forced aligner: Trainable text-speech alignment using kaldi. In *Interspeech 2017*, pages 498–502, 2017. doi: 10.21437/Interspeech.2017-1386.