UNIVERSITÀ degli STUDI di CATANIA

Department of Electrical, Electronic and Computer Engineering

MASTER'S DEGREE IN COMPUTER ENGINEERING

Few-Shots Learning Models for Classification and Localization of Bio-Acoustic Events

Accademic Year 2023-2024

Candidate: Aldo Barca

Supervisor: Prof.ssa Daniela Giordano

Co-Supervisor: Dr. Salvatore Calcagno

Co-Supervisor: Dr. Simone Carnemolla

ABSTRACT

The increasing demand for advanced bio-acoustic monitoring techniques has driven research into more efficient methods for classifying and localizing bio-acoustic events. This thesis is motivated by the DCASE 2024 international challenge, which highlights the need for robust solutions in this domain. Bio-acoustic events, generated by living organisms, offer valuable insights into their behavior and environmental interactions but require accurate classification and localization to be effectively analyzed.

Addressing the challenge of limited labeled data, this research applies Few-Shot Learning (FSL) techniques to bio-acoustic event detection. Specifically, it’s utilized a Prototypical Network combined with a dual loss function—triplet loss and cross-entropy loss—to enhance the model's ability to generalize from few examples. The study explores various encoders, including AST (Audio Spectrogram Transformer), Wav2Vec, and Convolutional Neural Networks (CNNs), to process and analyze bio-acoustic data represented as Mel Spectrograms.

The primary objectives of this thesis are firstly to assess the effectiveness of Few-Shot Learning approaches, particularly the Prototypical Network with a dual task and a consequential dual loss function, in improving the classification accuracy of bio-acoustic events; and secondly, to develop methods for precise localization of these events within audio recordings thanks to a 10 ms frame based system. The research involves a series of experiments to evaluate these methods, in consideration also to the DCASE 2024 challenge benchmarks, demonstrating the necessity of FSL to address data scarcity issues and improve bio-acoustic event analysis.

This thesis aims to contribute to the scientific community by providing a comprehensive analysis of the task specification and challenges, as well as offering also practical insights and innovative techniques. Moreover, the research proposes to advance understanding in this field and demonstrates how new and existing

techniques can be effectively utilized and refined together to address specific challenges.

ACKNOWLEDGEMENTS (Ringraziamenti)

Vorrei esprimere la mia più sincera gratitudine a tutte le persone che hanno contribuito direttamente e indirettamente alla realizzazione di questa Tesi.

Innanzitutto vorrei ringraziare la mia relatrice, la Prof.ssa Daniela Giordano per il supporto e le sue preziose indicazioni, per avermi motivato e incoraggiato ad entrare in un mondo complesso ma magnifico come questo. Senza di lei questa tesi non sarebbe mai nata.

Un ringraziamento di tutto cuore va ai miei due correlatori, il Dr. Salvatore Calcagno e il Dr. Simone Carnemolla, compagni in questa avventura, che con la loro disponibilità, il loro supporto incondizionato, il loro entusiasmo e le loro profonde conoscenze hanno permesso a me di migliorare e imparare sempre di più. Senza di loro questa tesi non sarebbe la stessa.

Il ringraziamento più grande va ai miei genitori, Enzo e Patrizia, che in questo percorso sono stati fondamentali. Sempre disponibili a ogni sacrificio pur di supportarmi e sempre vicini in ogni momento difficile, mi hanno permesso di sopportare ogni situazione complessa e superare ogni ostacolo. Dedico a loro questo risultato.

Un ulteriore grande ringraziamento va ad Andrea La Rosa, amico insostituibile che oltre che spronarmi continuamente a dare il massimo, mi ha permesso di discutere di questo elaborato dandomi preziosi consigli che mi hanno permesso di vedere molti aspetti sotto una luce diversa.

Ringrazio di cuore Giovanni, Francesco, Alessandro e Vincenzo. Colleghi e poi amici che mi hanno accompagnato in questo percorso. Le mattine degli esami con voi rimarranno un ricordo indelebile.

Un ultimo e sentito ringraziamento va a Giulia, Flavia e Alberto, amici sempre presenti che mi hanno sempre dato forza senza farmi mancare mai la loro compagnia e supporto, soprattutto nei momenti difficili.

INDEX

Introduction

The study of bio-acoustics has emerged as a crucial field for understanding and monitoring ecosystems, wildlife behavior, and environmental changes. By analyzing bio-acoustic events, such as animal vocalizations and environmental sounds, researchers can gain valuable insights into biodiversity and habitat conditions. However, the vast amount of acoustic data generated in natural environments presents a challenge for traditional machine learning models, which require inevitably large labeled datasets to achieve high performance.

This research work is driven by the challenges of the task 5 of the DCASE Challenge 2024, focused on advancing bio-acoustic event classification and localization through the necessity of handling the data scarcity instrinsic in this field. This challenge highlighted the limitations of existing methods and underscored the need for innovative solutions capable of performing well with limited training data. In this regard, few-shot learning has emerged as a promising approach to address these challenges, offering the ability to make accurate predictions from a minimal number of examples by leveraging prior knowledge and adapting learned patterns from related tasks.

In response to this challenge, this thesis explores the application of few-shot learning techniques to the classification and localization of bio-acoustic events, with a particular focus on Prototypical Networks. Prototypical Networks are a type of metric-based few-shot learning approach that have shown promise in various domains by learning a metric space where classification can be performed based on distance to prototype representations.

By integrating prototypical network with different types of encoders and multiple advanced pre-processing tecniques, this research aims to advance the state-of-the-art in bio-acoustic event classification and localization, offering valuable insights for researchers and practitioners in the field. The outcome of this research aim to contribute to the development of more efficient and scalable bio-acoustic monitoring systems, capable of learning with few examples and accurately classifying and localizing events even in challenging environments.

1. Audio models

In this first chapter, the main focus is to address concisely the technologies used for this thesis project. A secondary objective of the chapter is to discuss machine learning with regards to audios. Additionally, will be discussed the concept of few-shot learning and the state of the art in this regard.

1.1. *Audio Machine Learning*

Machine Learning is more popular than it has ever been and with increased popularity comes also the necessity of different and more complex application domain. One of the most interesting and studied of these domains is for sure the audio one.

With the help of advanced algorithms, machine learning models can analyze and interpret complex audio data, leading to significant improvements in tasks such as speech recognition, audio classification, and sound synthesis. These applications have found their way into various industries, including entertainment, healthcare, and security where they enhance user experiences and enable real-time audio analysis, providing innovative solutions for problems like noise reduction and audio forensics. Moreover, these models can be used in popular and discussed domains like speech recognition, speech transcription and, as discussed in this thesis work, in audio classification and event detection. [1]

1.1.1. *Audio Machine Learning challenges*

Applying machine learning to audio data introduces several challenges that can affect the performance and reliability of models. The most interesting and complex ones to handle are:

- Data Quality and Preprocessing: Audio data can be noisy and inconsistent, requiring extensive preprocessing to remove artifacts, normalize volumes, and handle missing or corrupted data. The quality of the input data directly impacts the effectiveness of machine learning models. [2]
- Feature Extraction: Raw audio signals are high-dimensional and complex, making feature extraction a crucial step. Identifying and extracting meaningful features which represent relevant aspects of the audio, such as pitch, timbre, or rhythm, is really hard but at the same time crucial to achieve high performances.
- Data Scarcity: High-quality labeled audio datasets are often limited, particularly for specialized tasks or languages. This scarcity can hinder model training and generalization, making it difficult to develop robust models without large amounts of annotated data. In the application scenario where the dataset is already available, this is a no problem, but in others scenarious this scarcity can lead to the doom of a model’s performance.
- Variability in Audio Data: Audio data can vary widely due to differences in people voices, recording conditions, and background noise. This variability can make it challenging for models to generalize across different environments and speakers.
- Temporal Dependencies: Audio signals have temporal dependencies that need to be captured for tasks like speech recognition or music analysis. Modeling these dependencies requires sophisticated techniques, such as recurrent neural networks (RNNs) or transformers, which can be computationally intensive.
- Real-Time Processing: For applications as live transcription and real-time audio effects, models must process audio data with minimal latency. Balancing real-time performance with accuracy can be challenging and may require optimization of both algorithms and hardware.
- Generalization and Overfitting: Audio Machine learning models as every models in general can overfit to the training data, leading to poor performance on unseen audio samples. In the audio field the overfitting scenario can be even more common, given the more complex representation

necessary to capture an audio features respect to an image features. Ensuring a model can generalize well requires careful validation and techniques like data augmentation.

- Ethical and Privacy Concerns: Audio data can be sensitive, raising ethical and privacy issues, especially when dealing with personal or confidential information. Ensuring that models are used responsibly and that data privacy is maintained is crucial to maintain a respectful ethic.

To solve most of these hard and stimulating challenges it’s necessary to understand at a deep level the specific scenario in which the task is done for. Moreover, the difficulties discussed until now need a more analytical approach to the problem of the representation with regards to the audio field.

1.1.2. *Representation Learning*

Representation learning is a subfield of machine learning that focuses on automatically discovering and extracting meaningful features or representations from raw data. Unlike traditional methods where features are often hand-engineered by experts, representation learning seeks to learn these features directly from the data, making it particularly effective in complex domains such as the ones involving audios discussed before.

In particular in the audio Machine Learning field, representation learning focuses on automatically extracting and learning meaningful features from raw audio signals, which are crucial for various audio-related tasks. This approach leverages sophisticated algorithms and neural network architectures to transform complex audio data into more manageable and informative representations.

For instance, in speech recognition, representation learning helps in transforming raw audio waves into features that can be effectively used by machine learning models to transcribe spoken words. Similarly, in music genre classification, models learn representations that distinguish between different musical styles by capturing patterns in audio features.

Another example even more fitting to this thesis is the event detection domain. In fact, representation learning plays a crucial role in event detection by automatically learning features from data that are useful for identifying and categorizing specific events. In event detection, whether in audio, video, or other data streams, the ability to extract meaningful and discriminative representations from raw input is essential for accurate identification and classification.[3]

1.2. Audio Processing

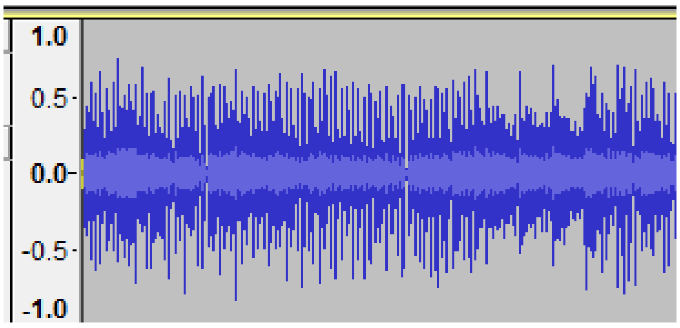
Audio processing is a field dedicated to the analysis, manipulation, and synthesis of audio signals. It encompasses a range of techniques and technologies aimed at improving, transforming, or extracting useful information from audio data.

In a different way from a simpler domain like computer vision, audio machine learning has the necessity to process the audio. This is also necessary to implement the already discussed representation learning, where important and relevant characteristics (features) are derived from raw data using complex audio manipulation operations.

Moreover, an audio needs additional operations to be converted into a suitable format to be then processed by the different state of art machine learning models (usually it must be represented using vectors or tensor). The vast majority of the most used pre-trained models are not created to support raw audio. Instead, they need the raw audio to be converted into spectrograms. However, Wav2Vec, an important model used in this work uses as input raw audios. These operations in implementation phase are usually made using python libraries as torch audiovision or librosa.

1.2.1. *Raw Audio*

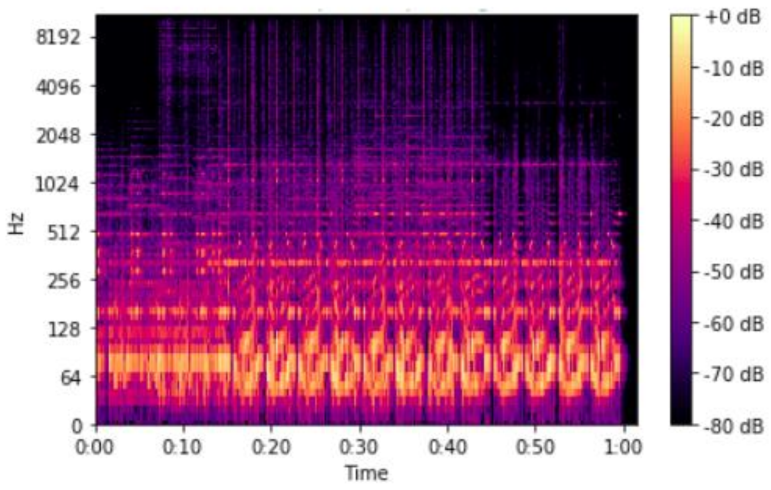
A raw audio as the name suggests is a non compressed, non elaborated audio signal represented as a time-domain waveform. This is done by plotting in the x-axis the time and in the y-axis the amplitude of the audio signal. Each point on the time-axis represents a sample of the original audio. The sample rate in machine learning audio task is usually 16 KHz or 22.05 KHz based on the type of sounds the model interacts with. For example if the model is based on a bird songs classification task, the frequency must be 22KHz to be capable of capturing the frequency of the event. . The sample rate defines also the quality of the signal representation. Given Nyquist theorem it must be at least 2 times the max frequency present in the audio signal.



[4]. Visualization of a raw audio.

1.2.2. *Spectrograms*

A spectrogram is a visual representation of the intensity of frequencies in an audio signal over time. In particular from left to right it shows how characteristics of the signal change over time. This representation is created by plotting in the x-axis the time. The y-axis represents the frequencies of the audio signal, with low-frequency sounds on the bottom and high-frequency at the top. There is another "hidden" dimension, the color of the spectrogram which represents different amplitudes of the frequencies at a given moment. The more intense the color (brighter or darker depending on the color scale used), the higher the energy of the corresponding frequency at that time.

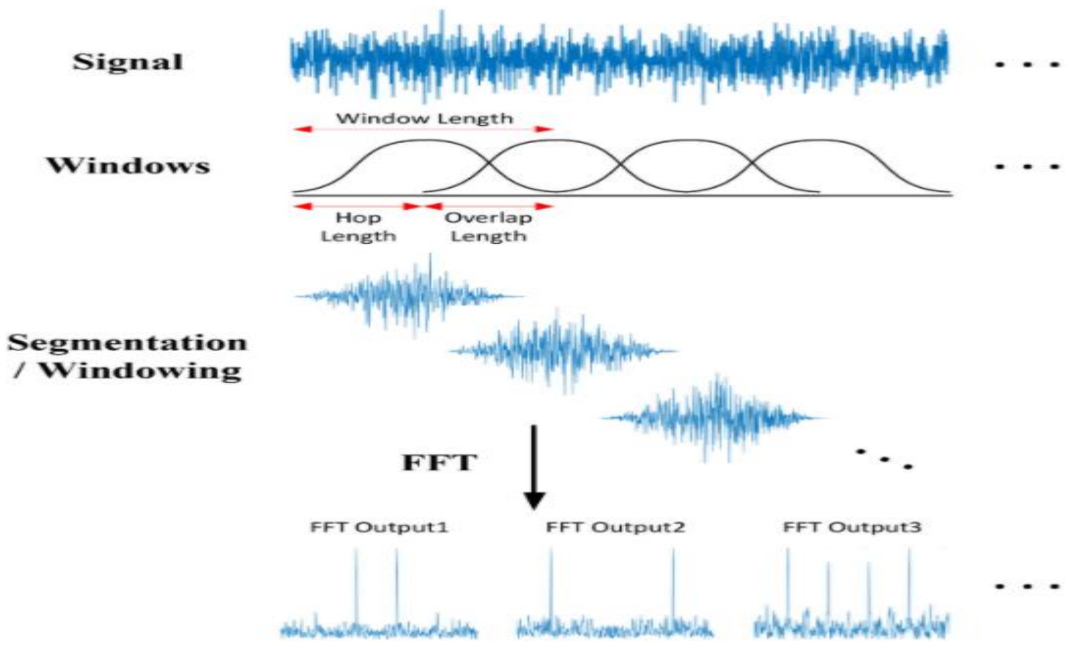


1.2.3. *Spectrogram calculation*

The spectrogram is generally calculated by dividing a continuous audio signal into time windows (segments) and then applying the Short Time Fourier Transform (STFT) or a sequence of band-pass filters to each window. This produces a frequency spectrum for each time window, and the sequence of these spectra generates the spectrogram.

The key point of the spectrogram creation is to decide two key parameters of the STFT operation which leads to the effective spectrogram:

- Window length: The length of the fixed intervals in which STFT divides the signal. The choice of a considerable windows gives a better distintion between frequencies but at the same time it’s harder to identify fast changes in the signal (for example short events which start and ends in few milliseconds).
- Hop length: The length of the non-intersecting portion of window length. The choice of a huge hop length entails less overlapping windows and therefore less “redundant” data to compute. At the same time it means a greater chance of aliasing and it makes harder to catch fast changes in the signal.



[2]. Visual representation of a spectrogram creation.

1.2.2. *Mel Spectrogram*

Another key component of the audio processing is the concept of Mel Spectrogram. From a matemathical point of view the Mel scale is the result of a non-linear transformation of the frequency scale. The objective of this different scale is to represent how perceptive human ear really is. Therefore, if a human can clearly understand the difference between two sounds, then these sounds would be far away in the Mel Scale. This is better than the Hz scale where the difference between 500-1000 Hz and 7500-8000 Hz is equal, but to the human ear the difference between 500–1000 Hz is quite noticeable. At the same time for 7500–8000 Hz the difference is barely understandable.[5].

Thanks to the more aligned to human mode, the Mel Spectrograms are extremely effective in speech recognition and in all application where a human would classify well. At the same time another important advantage of this representation is computational efficiency, even better than linear spectrograms thanks to the decreased number of frequency bands to process and store. Last but not least Mel

Spectrograms can be more robust to noise and distortions due to the compression of frequencies and the closer to human representation of sounds

However, Mel representation fails in every situation where the application domain features high frequencies, due to the valuable details lost in these frequencies. Moreover, Mels feature a lack of fine details, especially in scenarious requiring advanced acoustic diagnosis and also an increased complexity of interpretation of the spectrogram.

1.3. State of the Art Models for Audio-Based Machine Learning

Thanks to the growing interest and increasing number of studies in the scientific community, in the last years new model specialized in audio tasks are continuously increasing in number and more importantly quality. Based on the specific application lots of different and complex models are avaible for all the user. Moreover, every model has some key differences which maximize its performances in a specific scenario.

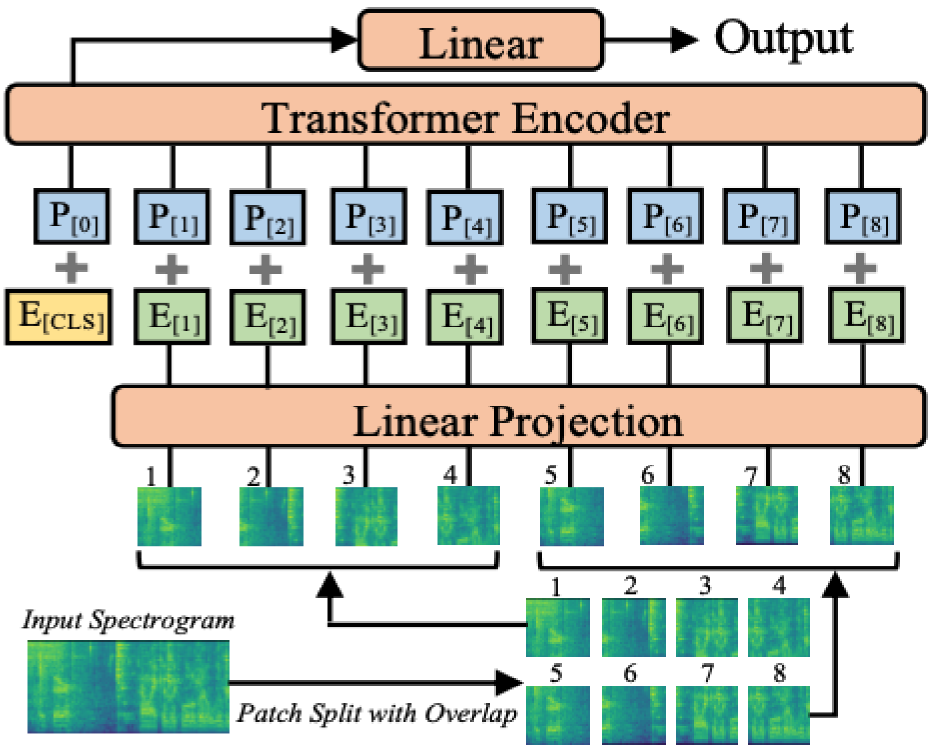
To achieve a higher grade of completeness must be taken as a standpoint that classical convolutional neural networks are still a valuable asset for a considerable amount of audio tasks. In fact, even if it’s hard to consider them the true state of the art in this regard, their ability to extract local features from time-frequency representations like spectrograms made them extremely effective. Despite the advancements of new architectures, CNNs remain highly practical, especially in scenarios where computational power is limited or where the simplicity of the architecture is an advantage which must be taken in consideration. This can be seen in scenarious like real-time applications or in cloud computing applications on edge devices, where computational efficiency and energy consumption are crucial aspects. Moreover, CNNs can also be used in combination with other models, such as RNNs or transformers, to leverage the advantages of convolution (local feature extraction) alongside sequential or long-range modeling capabilities.

In this paragraph the objective will be to have a brief compendium of all the most interesting state of the art audio Transformer models, with a particular emphasis on the type of task the model is suited for.

1.3.1. Audio Spectrogram Transformer (AST)

As the name suggests, AST is a Transformer model developed to work with spectrograms. This fundamental state of the art advanced deep learning model is specialized for audio classification tasks, therefore making it a first choice for tasks as environment audio classification and detection of acoustic events (events can space from identifying animal sounds in wildlife to monitoring or detecting anomalies in industrial equipment based on sound patterns).

Moreover, AST's ability to handle spectrograms as images makes it compatible with models used in other domains, facilitating cross-modal applications like audio-visual speech recognition or video captioning. [6]



Linear
Output
Transformer Encoder
$P_{[0]}$ $P_{[1]}$ $P_{[2]}$ $P_{[3]}$ $P_{[4]}$ $P_{[5]}$ $P_{[6]}$ $P_{[7]}$ $P_{[8]}$
$E_{[CLS]}$ $E_{[1]}$ $E_{[2]}$ $E_{[3]}$ $E_{[4]}$ $E_{[5]}$ $E_{[6]}$ $E_{[7]}$ $E_{[8]}$
Linear Projection
1 2 3 4 5 6 7 8
Input Spectrogram
Patch Split with Overlap
1 2 3 4
5 6 7 8

[7] AST model architecture graph.

Audio Spectrogram Transformer has demonstrated superior performance over traditional CNN-based models in many audio classification tasks. Its ability to capture both local and global dependencies in spectrogram data leads to more accurate classifications. These dependencies are crucial in task where the audio datas have a sequential nature, as speech recognition or event detection. Moreover, the transformer architecture allows AST to scale efficiently with the size of the dataset. As more audio data becomes available, AST can leverage this to improve its performance further, which is beneficial in applications with large datasets.

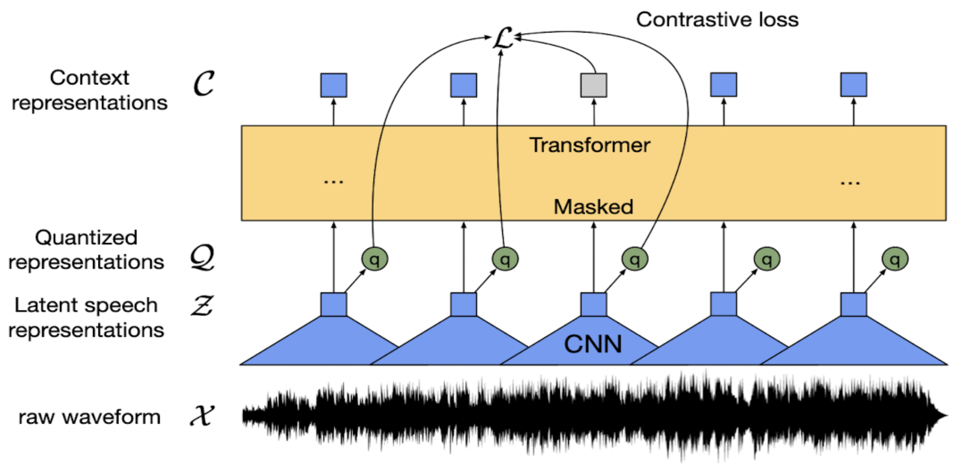
Even though AST is a great standpoint for audio transformer models, as common in many fields including machine learning, from its strength points come also its weakpoint. In fact, AST's transformer architecture, while powerful, is computationally demanding. It requires significant resources for training, including high memory and processing power, making it less accessible for small-scale applications or those with limited computational resources.

Additionally, a complex model like AST needs large datasets to reach its full potential and that's lead also to the need of a substantial amount of labeled audio. The required high dimensional size of the dataset makes it impossible to use an AST model and train it from scratch in scenarious like few/zero- shots learning. Therefore, leading to the inevitable use of as a pre-trained model. Lastly but not least due to its large number of parameters, AST is at a higher risk of overfitting in particular when trained on smaller datasets, needing an order of magnitude lower learning rate than usual.

1.3.2. Wav2Vec2

Wav2Vec2 is a self supervised model designed for Speech Recognition task which uses as input raw audio data. It represents a significant advancement in the speech recognition field by enabling models to learn useful representations from audio without relying heavily on large amounts of labeled data.

In particular, Wav2Vec shines in Automatic Speech Recognition scenarios where there are low-resource languages or domains where labeled data is scarce. Wav2Vec can also be pretraned on unlabeled audio, learning rich representations that can then be fine-tuned with a smaller amount of labeled data, achieving state of the art results. Moreover, thanks to its ability to learn from raw audio makes it adaptable to various languages and dialects, even those with limited available data. [8][9]



[10] Wav2Vec 2.0 model architecture graph.

Wav2Vec2 has its most notable feature the self-supervised learning approach which let it be pre-trained on vast amounts of unlabeled audio data, which is widely available, and then be fine-tuned with a relatively small labeled dataset. This reduces the dependency on large, expensive labeled datasets. Additionally in the Automatic Speech Recognition field it has set new benchmarks in ASR, particularly in low-resource settings. It outperforms traditional models by learning rich, contextual representations directly from raw audio, which results in more accurate transcription, especially in challenging conditions like noisy environments.

On the other hand, as discussed before regarding AST, Wav2Vec is also a complex model which requires significant processing power and memory, making it less accessible for small organizations or for applications where computational resources are limited. At the same time the performance of Wav2Vec stands in the quality of the pre-trained data. In fact, if the pre-training data is not representative of the target application, the model may not perform as well and therefore the right pre-trained model is required during fine-tuning. Lastly, fine-tuning Wav2Vec 2.0 requires careful adjustment of hyperparameters and a good understanding of the target task. Overfitting can occur if not properly managed, especially when fine-tuning on small labeled datasets.

1.4. Few-shots Learning

Few-shot learning is a machine learning paradigm where a model is trained to generalize from a small number of examples, typically much fewer than what is required by traditional learning algorithms. In traditional supervised learning, models are trained on large datasets with many labeled examples to learn the underlying patterns in the data. However, in real-world scenarios, collecting and annotating large amounts of data can be expensive or impractical, especially for specialized tasks.

Few-shot learning aims to address this challenge by enabling models to learn from a few examples or even a single example per class. This is particularly useful in scenarios where obtaining large amounts of labeled data is difficult, such as in medical imaging, robotics or natural language processing.

Moreover, by enabling models to generalize from limited examples, few-shot learning techniques have the potential to make machine learning more accessible and applicable to a wide range of real-world problems.[11]

In the context of Few-Shot learning are introduced three specific concepts used to describe the problem itself:

1. **Support set**: A subset of the dataset which contains a limited amounts of labeled examples (images, text or even audio events). These events are linked to a specific class in the dataset. The support set is used to provide the model the crucial informations and examples, to learn and generalize during training.
2. **Query set**: Another subset of the dataset containing usually the remaining portion of the dataset itself. The query set is composed of unlabeled examples that must be classified into one of the classes used in the support set. After an initial training in the support set, model's performance are evaluated based on the classification of the query set.

3. **N-way K-Shot**: The standard notation of Few-Shot learning. The N refers the number of classes in the support set and the K refers to the number of examples (or events in audio classification) associated for each class. [12]

There are several model and tecniques used for few-shot learning and dealing with data scarcity. The most significant ones are:

- Siamese Networks
- Transfer learning
- Meta-Learning
- Prototypical Networks
- Data Augmentation

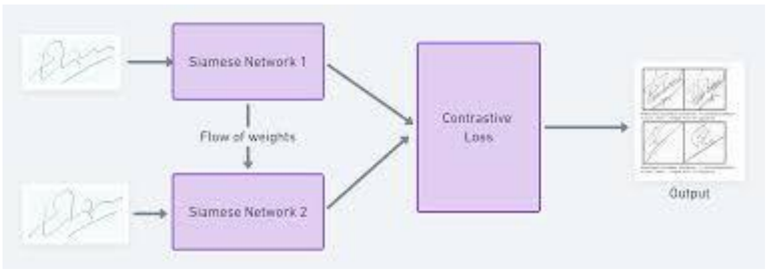
1.4.1. Siamese Networks

A Siamese neural network, sometimes referred to as a twin neural network, is an artificial neural network that operates on two different input vectors using identical weights to produce comparable output vectors. Typically, one of these output vectors is precomputed, establishing a baseline for comparison with the other output vector. This process is analogous to fingerprint comparison and can be more technically described as a distance function for locality-sensitive hashing.

Applications of similarity measures where Siamese networks are beneficial include recognizing handwritten checks, automatic face detection in camera images, and matching search queries with indexed documents. The most well-known use of twin networks is in face recognition, where precomputed images are compared with new images from sources like entry gates document verification. There are two main problems in face recognition: identifying a person face/ voice among many others (recognition) and verifying if a photo/audio matches the person claiming to be in the photo (verification). Although the networks used for both tasks might be similar, their implementations can differ significantly.

Siamese networks learn to create embeddings (representations) for input samples, using distance metrics to compare these embeddings for similarity-based classification. They measure and compare the similarity between two inputs, making them particularly useful when there are examples available for each class. In few-shot learning, Siamese networks are employed to learn a similarity metric between support set examples (labeled examples) and query set examples. [13]

The goal is to minimize a distance metric for similar objects and maximize it for distinct ones. This principle guides the loss function. The most common distance metric used is Euclidean distance.

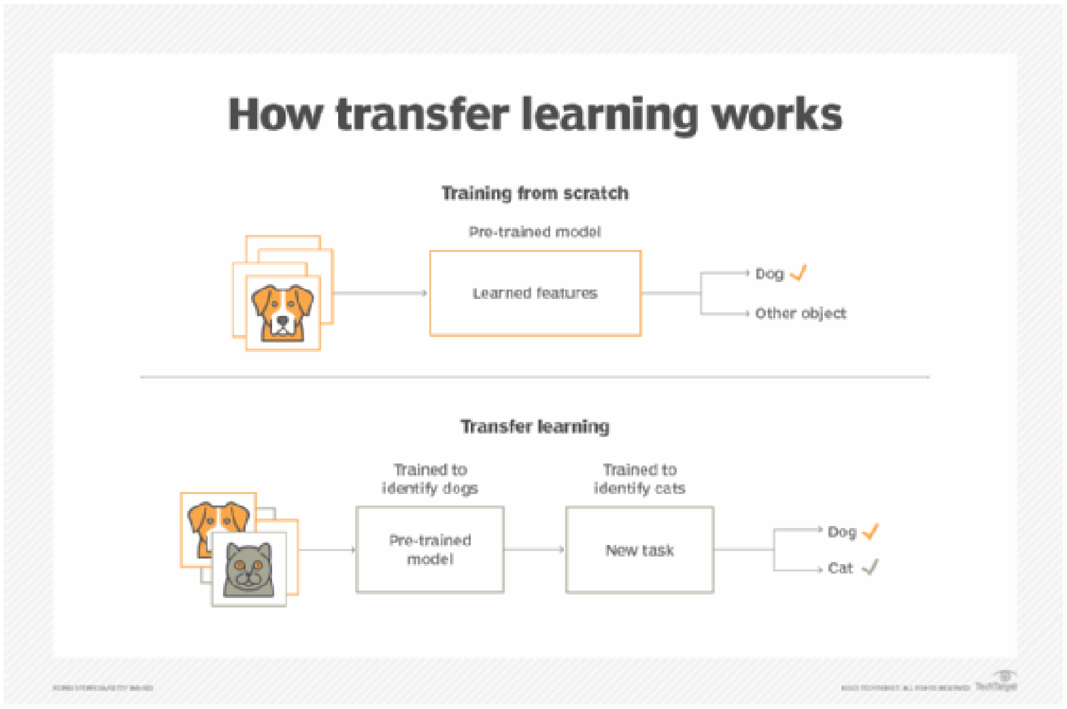


[14] Structural architecture of a Siamese Neural network.

1.5. Transfer and Meta learning

1.5.1. Transfer learning

Transfer learning (also referred as fine tuning) is a key machine learning method where an already developed model is used another time in a different task. This allows to train the "old" model with less datas compared to a new model created from scratch. Thanks to transfer learning it's possible to reduce greatly the amount of time and computational resources needed for the training, in particular for complex models as AST or Wav2Vec where the from scratch training would take a huge amount of time and resources. Moreover, transfer learning is a great technique in few-shot learning, where the amount of data which can be used during training is almost non existent.



Transfer learning leverages knowledge gained from solving one problem and applies it to a different, but related problem. This process is done taking a pre-trained model (usually trained on a large dataset for a related task) and fine-tuning it on a smaller dataset for a specific task. By doing this, the model can learn to generalize better and perform well even with limited labeled data.

In transfer learning, knowledge from a source task is used to enhance learning in a new task. When the transfer method reduces the performance of the new task, it's called a negative transfer. A major challenge when developing transfer methods is ensuring positive transfer between related tasks while avoiding negative transfer between less related tasks. When knowledge is transfered from one task to another, the original task's characteristics are usually mapped onto those of the other task to specify correspondence. Traditionally, this alignment is provided by a human, but evolving methods are emerging that automate this mapping process.

Transfer learning can be executed through various methods. One approach involves identifying a related task, denoted as Task 2, that possesses ample labeled data transferable to the new task. The model is subsequently trained on Task 2, providing it with a foundation for addressing its primary task, Task 1.

Alternatively, transfer learning can leverage a pre-trained model, simplifying the process by utilizing an already trained model. The pre-trained model should have been trained on a large dataset for a similar task as Task 1. These models can be sourced from online repositories or other developers who have shared their work.

A third strategy, is the use of and already representation learning from another model to discern the most significant features for Task 1 (the main task), thereby establishing a representation of the task. While features are traditionally crafted manually, deep learning autonomously extracts features. Data scientists then select the pertinent features to inclide in the model. This learned representation can also be applied to other tasks.[14]

When using transfer learning, it is important to choose the most suitable strategy, and there are three key points to consider:

- The first and most crucial question is the identification of which part of the knowledge should be transferred from the source to the target in order to address the target task's problem.
- Another important point is when the transfer should be made. There can be scenarios where the application of Transfer Learning methods may lead to a degradation of results. In reality, these performances depend on the similarity between the source and target domains and tasks
- Lastly must be choose how is the transfer going to be done. Once the "what" and "when" questions have been addressed, the most appropriate Transfer Learning technique for the problem at hand can be rightly identified.[15]

1.5.2. Meta Learning

Meta-learning, with its focus on acquiring the ability to adapt to new tasks, plays a pivotal role in tackling the obstacle presented by scarce labeled data in few-shot classification scenarios. This enables models to leverage previous knowledge garnered from different tasks to operate efficiently with minimal instances for unfamiliar tasks.

Meta-learning in general means learning about learning. The focus of meta-learning is to create models and algorithm where the main objective is to adapt quickly to new tasks and environments, even with minimal amount of training data as given in few-shot learning. This approach can achieve new heights in learning new concept as fast as possible and with few amounts of examples, as a human would do, and even in more complex scenarious where the datasets aren't huge or detailed. [16]

The key idea is to let meta learning models to improve the learning ability over time, using two different level in learning: an inner and an outer level:

- The *inner level* refers to the learning of a new single task. The agent tries to learn the associated concepts from the dataset. This learning often involves only a small amount of data for each task, as the model leverages the knowledge gained from the outer level to quickly adapt.
- The outer level (also called meta-learning level) involves learning across tasks or domains. The model learns patterns or strategies from a distribution of tasks, which it can then use to adapt to new tasks more efficiently.

In regular supervised learning the key point is to obtain the globally optimal parameters ϑ, where the objective is to find the right ϑ which minimize:

$$\vartheta = \underbrace{\arg min}_{\vartheta} \text{ loss\_function}(\vartheta_0)$$

In general, directly finding the best ϑ is computationally hard or even unfeasible. It's possible however to approximate ϑ using pre-defined knowledge $\acute{\omega}$. This parameter includes the initial model parameters ϑ_0, the choice of an optimizer and the learning rate schedule.

In meta learning the main objective is to find the best ώ instead of assuming ώ as pre-given as in classical deep learning. Obtaining the best ώ is key to implement a model able to learn new tasks as quickly as possible. The right parameters are heavily influenced by the set of tasks the model must do and their frequencies in percentage.

1.5.2.1 Meta-Learning vs Transfer Learning

Considered the transfer learning discussed before, it's entirely possible to transfer knowledge of previous tasks to new, unseen tasks. That's why over time, the line between Transfer Learning and Meta-Learning has become increasingly blurred. A defining characteristic of meta-learning techniques, however, is their meta-objective, which explicitly targets performance optimization across a task distribution. This meta learning modality is pivotal in the research for unsolved and stimulating challenges like the few and zero shot learning discussed in this work.

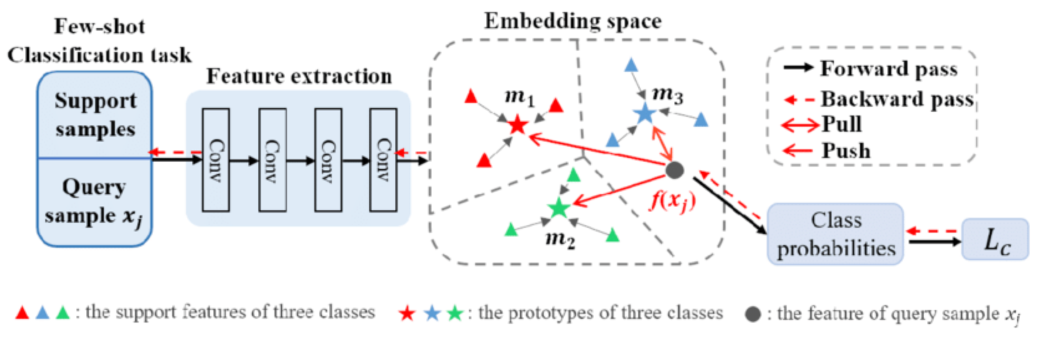
1.6. Prototypical network

Prototypical Networks are a type of neural network architecture specifically designed for the already discussed few-shot learning. Introduced by Jake Snell, Kevin Swersky, and Richard Zemel in their 2017 paper "Prototypical Networks for Few-shot Learning" [18].

A classical Prototypical Network uses a different way of training. Instead of doing a classical epoch based training, a new concept called episode is introduced. An episode is designed to mimic the few-shot task by subsampling classes as well as data points. The use of episodes makes the training problem more faithful to the test environment and thereby improves generalization.

The key difficulty of a few-shot learning task is the overfitting. The severely limited amount of data leads to the necessity of a model to learn not only thanks to its few-shots but also how to make the best representation possible. In this regard the prototypical network is based heavily on the concept of embedding and on the robustness of this embedding. The core idea of Prototypical Networks is to map input data (images, text, or audio) into an embedding space using a model as AST, Wav2Vec or a simple CNN called encoder. In this space, data points from the same class should cluster together.

The prototypes, representation of each of the classes, is the mean of the embeddings (feature vectors) of all examples in that class. These prototypes are then used to classify new unseen examples by the use of a distance metric like Euclidean distance, cosine distance or Manhattan distance. Each type of distance shines based on the type of embeddings it computes the distances. The Cosine one for example is excellent for natural language processing tasks where angular similiarity is a key factor. The nearest prototype gives the predicted class for the example.



[19] Architectural structure of a generic Prototypical Network.

In particular the prototypes are compared to the embeddings of the new samples. These new examples are what’s called query set. The support set is the part of the dataset used to create the prototypes for the classes.

Considering the intrinsic effectiveness in learning from few data, Prototypical Network are heavily used in scenarious like the evergreen Few-Shots learning and speech and audio classification tasks such as identifying new speakers or recognizing specific sound events with limited labeled data. Moreover, its capability of generalization with a data scarcity scenario makes these Networks to be well suited for meta-learning scenarios such as robotics, personalization systems, and other adaptive technologies.

One of the most interesting advantage of this flexible type of model is the great scalability to a large number of classes, as the complexity of the classification decision scales linearly with the number of prototypes rather than the number of individual examples.

On the other hand, its drawback stands in a high dependance on the quality of the embedding space. If the embeddings do not capture the relevant features well, the prototypes may not accurately represent their classes, leading to poor performance. Lastly even if the concept as a baseline is simple and clear, the episodic approach involves a greater number of complications and necessary choices and criteria to be effective, making the training phase and algorithm to beeven more complex to handle than most of the classic epoch based training procedures.

2. Few-shot Bioacoustic Event Detection

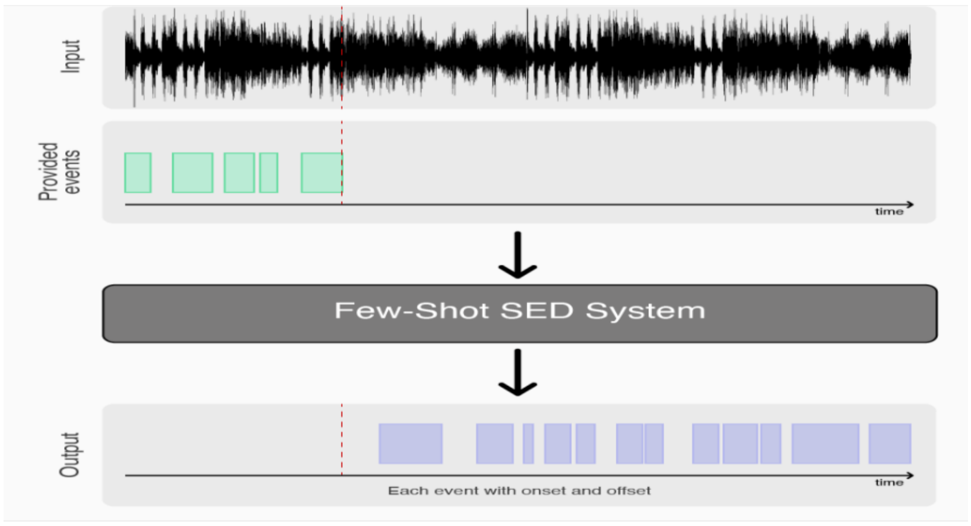
The focus of this thesis work is to explore the world of the few-shot learning for acoustic event using as a baseline the challenge 5 of the DCASE Challenge 2024, Few-shot Bioacoustic Event Detection. As the name of the task suggests, the challenge main theme is sound event detection in a few-shot learning setting and in particular for animal (mammal and bird) vocalisations.

The main objective of the challenge is to create a model able to detect specific animal sounds in an audio. Furthermore, another fundamental criterion is the obligation to train the model with a few-shot supervised learning algorithm.

The reason for the technical committee to give a so hard constraint is explicit referred in the official documentation for the task: “Few-shot learning is a highly promising paradigm for sound event detection. It is also an extremely good fit to the needs of users in bioacoustics, in which increasingly large acoustic datasets commonly need to be labelled for events of an identified category (species or call-type in this case), even though this category might not be known in other datasets or have any yet-known label.” [20]

2.1. *Task specification and explanation*

The task of the challenge presents itself as an event detection task. From a pragmatic point of view, it requires a model which can at the same time given only k shots (k=5 in this case) to classify an entire audio, locating all the instances of a given class represented by the k shots. Moreover, the challenge team requires the partecipants to use as the primary metric score in inference phase for the task the F1 score, which is a really strict metric.



[20] High level representation of the task provided inputs and required output.

The final objective is to create and train a model capable of generalizing from only 5 event examples and which can be able to detect if an event is of the same type or not. This complex task requires the model to be able to extract from few samples the meaningful informations which distinguishes that particular animal class sound from the others. Therefore, the model must have an incredible learning capability and must use effective strategies based on transfer learning or meta-learning. The task is not difficult only for what has been said so far, there are other intrinsic difficulties whose cannot be ignored.

One among all the intrinsic variability of bioacustic sounds. In fact, bioacustic events can vary significantly due to:

1. Differences within the same species, for instance the same bird or hyena can produce substantially different songs depending on the specific context of the event.
2. Different environmental conditions caused by recording conditions like background noise, environmental interference, and distance from the sound source.

3. Many behavioral factors. In fact, animals can alter their sounds depending on specific behaviors, such as mating, territorial defense, or warning communication.

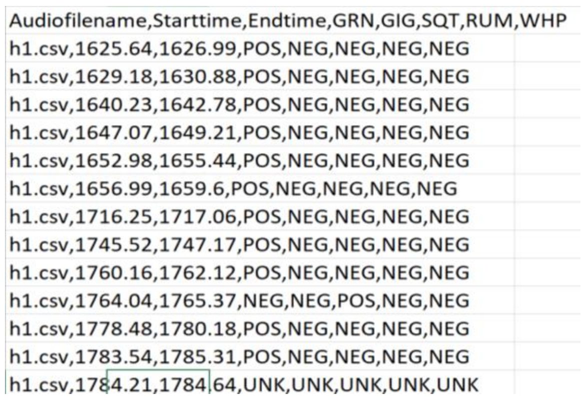
An additional challenge for the task is the necessity to develop specialized models. Conventional machine learning methods may easily fail with these few-shot conditions, necessitating the development of specialized models such as the already discussed prototypical networks, or siamese networks designed to perform well with few examples.

Last but not least the required complexity to handle the temporal dinamics for events. The temporal dynamics of bioacoustic events add another layer of complexity, as the model needs to understand when an event starts and ends, and how it evolves over time. Detection requires not just identifying the sound but also correctly segmenting it temporally which involves others crucial aspects to keep in mind.

2.2. Dataset specification and explanation

The challenge team, along with the challenge specifications, has provided a dataset to all participants. The dataset consists of two splits, one training and one validation/inference set. [20]

The training set consists of five sub-folders deriving from a different source, each for a different macroclass of animals (for example hyenas). Along with the audio files, multi-class annotations are provided for each. These annotations are provided in a csv file with the same name of the audio, containing for the audio all its bioacustic events contained in it. For each event is provided the start and endtime in seconds, plus the POS label for the class actually in the event and the NEG label for all others class.

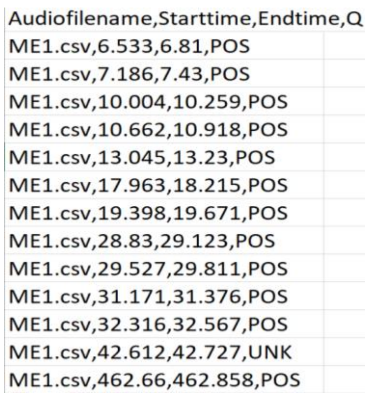


```
Audiofilename,Starttime,Endtime,GRN,GIG,SQT,RUM,WHP
h1.csv,1625.64,1626.99,POS,NEG,NEG,NEG,NEG
h1.csv,1629.18,1630.88,POS,NEG,NEG,NEG,NEG
h1.csv,1640.23,1642.78,POS,NEG,NEG,NEG,NEG
h1.csv,1647.07,1649.21,POS,NEG,NEG,NEG,NEG
h1.csv,1652.98,1655.44,POS,NEG,NEG,NEG,NEG
h1.csv,1656.99,1659.6,POS,NEG,NEG,NEG,NEG
h1.csv,1716.25,1717.06,POS,NEG,NEG,NEG,NEG
h1.csv,1745.52,1747.17,POS,NEG,NEG,NEG,NEG
h1.csv,1760.16,1762.12,POS,NEG,NEG,NEG,NEG
h1.csv,1764.04,1765.37,NEG,NEG,POS,NEG,NEG
h1.csv,1778.48,1780.18,POS,NEG,NEG,NEG,NEG
h1.csv,1783.54,1785.31,POS,NEG,NEG,NEG,NEG
h1.csv,1784.21,1784.64,UNK,UNK,UNK,UNK,UNK
```

Image 1: Contents of the multi-class annotation h1.csv file for the training phase. Each row describes an event contained in the audio h1.wav.

The validation set consists of two sub-folders deriving from a different source each, with a single-class (class of interest) annotation file provided for each audio file. The annotation file for the validation set has a similar structure compared to the annotation file for training, but with the difference that for each audio is annotated only a single class, the one which is required to be localized for the audio. No informations are provided about the content of the validation set or the identity of the animal required to be localized in the query set of the corresponding audio. This is done to ensure the partecipants can use the validation set even for inference.

The few-shots nature of the task requires to use the first k events as the support set for the task and then localize and classify the remaining part of the audio.



```
Audiofilename,Starttime,Endtime,Q
ME1.csv,6.533,6.81,POS
ME1.csv,7.186,7.43,POS
ME1.csv,10.004,10.259,POS
ME1.csv,10.662,10.918,POS
ME1.csv,13.045,13.23,POS
ME1.csv,17.963,18.215,POS
ME1.csv,19.398,19.671,POS
ME1.csv,28.83,29.123,POS
ME1.csv,29.527,29.811,POS
ME1.csv,31.171,31.376,POS
ME1.csv,32.316,32.567,POS
ME1.csv,42.612,42.727,UNK
ME1.csv,462.66,462.858,POS
```

Image 2: Contents of the annotation h1.csv file for the validation/inference phase. Each row describes an event contained in the audio h1.wav.

The evaluation set, provided later by the committee team, has a structure similar to the evaluation set. Therefore, it’s entirely possible to use the validation set to validate the model or simply use it for inference as a test set, making the evaluation set optional to use.

Due to the lack of lack of labelling for the query set of the audios in the evaluation set, the use of the validation set as inference set is even more necessary in this case where it hasn’t been submitted a model to the challenge directly and therefore, it’s impossible to perform a metric score as the F1 score.

For the validation/inference set are only annotated the events for the class for which detecting the events is required. Despite the presence of events from other classes, they remain unannotated and should not be predicted by the system.

2.3. Proposed solutions

In recent years, significant advancements have been made in regard to the DCASE challenge task 5, as demonstrated by two prominent approaches presented at the DCASE challenges in 2022 and 2023. These works have been unvaluable to understand the difficulties of the challenge and have offered interesting insights to develop an effective FSL system, capable of work in challenging audio environments.

2.3.1. Few-Shot Embedding Learning and Event Filtering

The team from iFLYTEK Research Institute and associated universities proposed a robust system for bioacoustic event detection, focusing on the enhancement of embedding learning under limited positive samples. Their approach incorporated both segment-level and frame-level embedding strategies, coupled with model adaptation techniques and an embedding-guided event filtering method. A notable innovation was their adaptive window length and shift strategy, which allowed for more precise event detection by adjusting the retrieval resolution based on the duration of positive support segments. Furthermore, the team emphasized the importance of reliable negative sampling, which was refined by selecting negative samples from specific segments of the audio, reducing the risk of misclassifying positive events as negatives. This system achieved an impressive F1 score of 74.4% on the validation set, showcasing the efficacy of their multi-faceted approach.[21]

2.3.2. Supervised Contrastive Learning for Pre-Training Few-Shot Systems

The team from IMT Atlantique and the University of Lorraine introduced a novel approach by using supervised contrastive learning (SCL) to enhance the feature extraction process for few-shot bioacoustic event detection. Their method involved training a feature extractor from scratch using only the official training data, without relying on any external datasets or pre-trained models. The core idea of SCL was to create a latent space where samples with similar class labels were clustered closely together, while those with different labels were pushed apart. This was achieved through a combination of data augmentations, including spectrogram mixing, frequency shift, and random cropping, among others. The resulting feature extractor was then fine-tuned on the few-shot validation and test sets, yielding a competitive F1 score of 63.46%. Notably, their approach did not require post-processing, making it simpler and more streamlined compared to other complex meta-learning frameworks. [22]

2.3.3. Approachs discussion

The iFLYTEK team's focus on embedding learning and event filtering, coupled with adaptive sampling techniques, provided a detailed and nuanced method for handling variable-length audio segments. On the other hand, the IMT Atlantique team's use of supervised contrastive learning represents a shift towards more generalizable and transferable feature extraction methods, particularly in scenarios where limited data is available.

The most interesting aspect of these works is highlighting the importance of embedding strategies, adaptive methods, and the potential of contrastive learning in advancing the state of the art in few-shot bioacoustic event detection.

3. Chapter 3

In this chapter the focus will be to retrace the approach used to handle the task. Moreover, will be discussed and enunciated the strategies and techniques used for the creation of a model able to make classification and localization of bioacustic-events given the hinderance of the few-shots scenario.

3.1. Task Approach

To handle a task of this magnitude, the first step was to thoroughly review the existing literature to understand what had already been done in this area. From this preliminary analysis, it became clear that a classical model and a traditional approach to the training phase would not be adequate for achieving satisfactory performances. In fact, using a conventional convolutional model, the resulting representation was unable to facilitate an effective event classification with the k=5 samples given as a support set. This situation is even more dangerous considering the audio nature of the samples, resulting in making it harder to obtain good embeddings. Additionally, using a convolutional encoder also resulted in losing the sequential nature that characterizes an animal’s call, highlighting the need to work with encoders that can leverage temporal order to determine whether an event is positive or not.

Another consideration that emerged especially from the need to localize the beginning and end of an event as well as classify it, was the division of audio into frames to transform the task into a binary classification problem (whether a frame contains a positive class or not). By doing so, it is possible to manage event localization, which will naturally be characterized by a sequence of positively classified frames.

However, the use of binary classification alone to distinguish frames cannot effectively handle the task due to the difficulty of creating an effective representation with few examples. Therefore, it was decided to manage classification not with traditional embeddings but through prototypes and consequently the use of a prototypical network regardless of the encoder type used. By employing a prototypical network, it is possible to create the prototype of the class that must be identified using the k=5 available events. Then, still using the frame-based system, the prototype of the target class (representing a specific animal) is compared with each frame of the remaining audio after the support set events. This is what will be considered the query set.

Moreover, to increase the capability of the model has been done another crucial consideration, the use of all the audio parts between the 5 positive support set events to obtain what can be called negative support set. This set can be used to create a negative prototype which consistute another way of classify effectively the frame. If a frame representation in the embedding space is closer to the negative prototype than it is to the positive prototype, then that frame can be classified as negative. To achieve this frame-based classification it has been necessary to choose an effective frame length, in this case 10ms.

From a training point of view, it has been clear that the solution more suited for a few-shots task is to use an episodic learning. For instance, if an audio has 4 classes in itself and for each class a support and query set, from that audio will be extracted 4 episodes. The episodes coinsist of the first 5 positive events, the positive support set, the negative support set which includes the audio portions placed between for that class and the query set. The query set is constituted by the remaining audio from the end of the last positive event to the end. Doing so with the whole dataset excluded the WMW macrocategory, results in a total of 54 episodes.

To effectively measure the model's performance, F1 score, Precision, and Recall will be used.

3.2. Dataset management and data operations

Having already discussed how the task was approached, it is now time to analyze how the data from the dataset has been obtained and has been managed to create the actual support and query sets to be used by the model. The process of obtaining the data was divided into various steps, each characterized by a Python function that allows for the transition from one data format to another. This approach was adopted to have the information available in multiple formats, regardless of the chosen data management criteria.

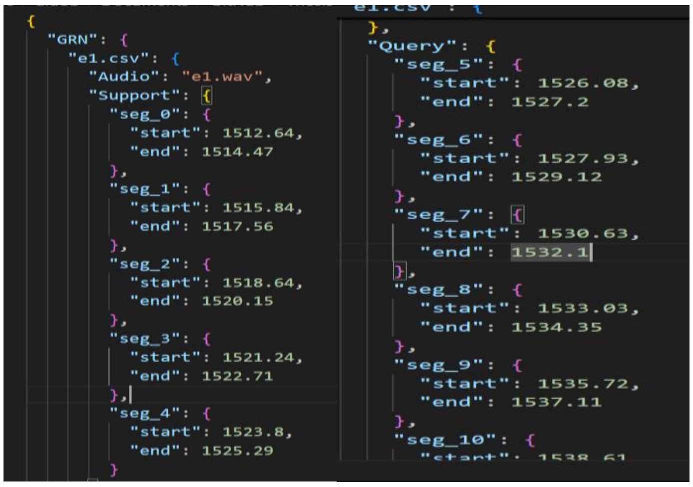
For clarity and completeness, the discussion of the data management steps will be divided into five subpoints:

1. Transition from CSV files to JSON files. In the JSON file, for each class there is info about the audio files in which the class appear. For each of these audios are included informations about the start and end times of positive events in the audio.
2. Extraction of all data from the JSON file to create a support set dictionary and a query set dictionary.
3. Complete labeling of all audio files.
4. Actual creation of the support sets and query sets for each episode.

3.2.1 JSON creation

Starting with the downloaded dataset from the challenge site, there are several folders, containing audio files (.wav format) and for each audio a corresponding .csv file. The objective of this code section is to successfully extract all the events for the classes contained in each audio. Once all the data from all the audios is obtained, then the datas are aggregated in class-based manner. It has been chosen to use the JavaScript Object Notation (JSON) for its readability and clarity, as well as for its lightness, which allows for rapid data extraction.

In particular the JSON contains a vector of classes, and for each class is associated every audio where it appears and each of its positive event in that audio. The events from each audio are divided in support and query set. The support set will contain the start and end time of the events for k=5 events. The query set will contain every other positive instance of the class for the audio.



```
{
  "GRN": {
    "e1.csv": {
      "Audio": "e1.wav",
      "Support": {
        "seg_0": {
          "start": 1512.64,
          "end": 1514.47
        },
        "seg_1": {
          "start": 1515.84,
          "end": 1517.56
        },
        "seg_2": {
          "start": 1518.64,
          "end": 1520.15
        },
        "seg_3": {
          "start": 1521.24,
          "end": 1522.71
        },
        "seg_4": {
          "start": 1523.8,
          "end": 1525.29
        }
```

```
"e1.csv": {
      },
      "Query": {
        "seg_5": {
          "start": 1526.08,
          "end": 1527.2
        },
        "seg_6": {
          "start": 1527.93,
          "end": 1529.12
        },
        "seg_7": {
          "start": 1530.63,
          "end": 1532.1
        },
        "seg_8": {
          "start": 1533.03,
          "end": 1534.35
        },
        "seg_9": {
          "start": 1535.72,
          "end": 1537.11
        },
        "seg_10": {
          "start": 1538.61
```

Image 3: Data representation of the JSON structures for Support and Query set.

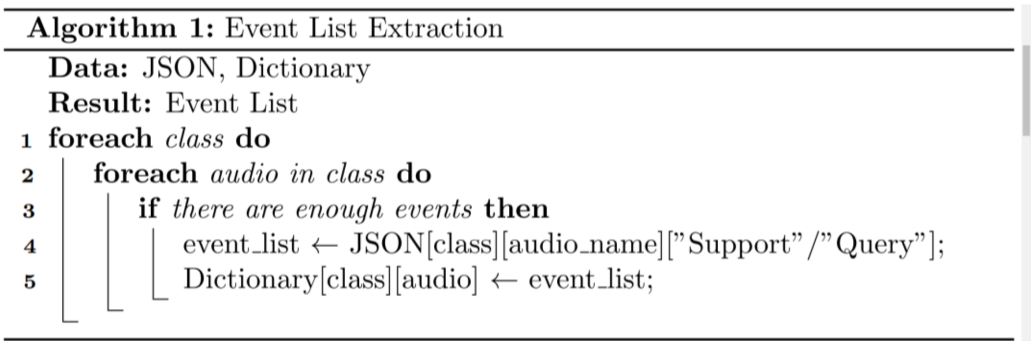
This operation has been done not only for the training datas (contained in the train_data.json file) but also for the inference data, contained in the val_data.json file. The substantial difference between training and inference/validation set is that in validation, for each audio is contained only a labelled class and that class is contained in exactly one audio. Therefore, for each class in the validation set only an audio is written. The support and query structure are the same as discussed so far.

3.2.2 Support and query set dictionaries creation

Once created the Json file, the next data operation consists of extracting the datas as codified in the JSON file and creating 2 different dictionaries, the support and query set. These dictionaries are, in fact, a 1:1 replica of the JSON datas and therefore contains datas for all the classes in the dataset.

During the dictionary creation, as an additional operation, it has been necessary to check if the audios have effectively k=5 support events to be considered. In the case where there aren't enough events, that particular class for the audio will not be included as a specific episode.

From a code point of view, the process includes iterating each class and for each class iterating each single audio where that class appears. Then if there are enough events the next and last step is to load from the JSON the list of events corresponding to the class plus audio combination. Here is reported the corresponding pseudocode:



Algorithm 1: Event List Extraction

```
Data: JSON, Dictionary
Result: Event List
foreach class do
    foreach audio in class do
        if there are enough events then
            event_list ← JSON[class][audio_name]["Support"/"Query"];
            Dictionary[class][audio] ← event_list;
```

3.2.3 Labelling

Once created a suitable dictionary containing all the events information for each class, the following step is to build a data structure which will contain the labels for all the 13 audios of the training set. The label structure is a list of lists, representing the list of frame labels for each audio. Therefore, each audio will have a different number of frames based on the duration. For instance, if the audio is 1h, then the number of frames and consequently labels for that audio is 3600s* frame_length_in_second. In this case the frame length is 0.01 s (10ms), therefore for 1 hour audio duration will corresponds 360 thousand labels, one for each frame.

From an implementative point of view the labels creation is done using the support and query dictionaries discussed before, plus an additional structure which contains all the names of the training data audio. Then will be created an empty data structure with all frame's label instantiated as 0. Using the event information in the dictionaries only the frames with a positive class will be labeled with labels different from zero. This choice has been made to improve labelling efficiency, knowing that in an audio over 90% of the time are unknowns sounds with a label=0 as default.

Using the training set of the challenge are present 22 different classes which would naturally correspond to a 0-21 labelling. However, the "unknown" sounds interspersed between the events can be crucial to handle the model's training. Therefore, the labelling goes from 0 (Unknown class) to 22 (22nd class).

To make the discussion as complete as possible, a pseudocode scheme that illustrates the algorithm will also be provided:

Algorithm 3 Audio Labels Creator Function

```
Input: support_set, query_set, audio, frame_length = 0.01
Output: labels
Initialize an empty list labels
Initialize class_dictionary with {'UNKNOWN': 0}
Load audio_path using librosa.load(audio_path)
Compute length_in_seconds as the total length of the audio divided by the
sample rate
Compute number_of_frames as the ceiling of the ratio between
length_in_seconds and frame_length
for each frame in number_of_frames do
  Append 0 to labels
end for
Initialize label_class to 1
for each event in support_set do
  Compute event_start, event_end
  Compute frame_start and frame_end as the integer division of event_start
  and event_end by frame_length
  for each frame from frame_start to frame_end do
    Set labels[frame] to label_class
  end for
end for
for each event in query_set do
  Compute event_start, event_end
  Compute frame_start and frame_end
  for each frame from frame_start to frame_end - 1 do
    Set labels[frame] to label_class
  end for
end for
Return labels
```

3.2.4 Support and query set creation

Once the support and query set structures and the label list have been created, it is now necessary to generate the actual data that will be put as input into the model. To start creating the datas, the first necessary step is to consider the episodic training approach used, therefore the data must be aggregated in a structure containing the 54 training episodes, even though using a single immense structure isn't a suitable option. In fact, to avoid slowing the training phase and to avoid memory problems has been deemed necessary the creation of a .pth file for each episode, containing the support and query set data as spectrograms or raw audio. This solution permits to make the training phase at least an hour shorter, having the datas already usable to the model. This is even more efficient if the data to be used is in spectrogram format, as the transformation into a spectrogram requires a lot of time and computational power.

The use of different data format (spectrograms or raw audio) is necessary to use different models which require different input. In fact, the implemented neural convolutional encoder requires spectrograms and transformer-based encoders requires the raw audio format as an input.

As discussed before (chapter 3.1), the support set used for the model isn't just constituted by the 5 positive events. Actually, all the audio fragments contained between the events will be considered as part of the support set. Naturally must be taken record of which data represent positive support events or the negative support frames.

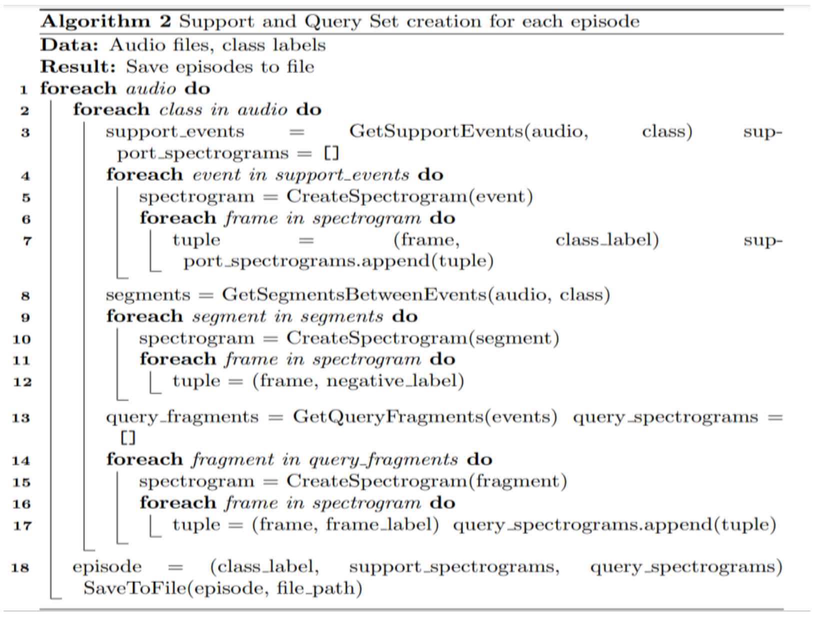
In particular for the spectrogram format for the classical convolutional encoder, has been used three different ways of handling the spectrogram creation:

1. Spectrogram created for each frame, with frame dimension of 10ms. Negative support set undersampled to have the same dimension as the positive support set.
2. Spectrogram created for each frame, with frame dimension of 10ms. Negative support set used fully without any undersampling.
3. Support set spectrograms created based on the mean duration of the 5 support events for that episode. Full negative support set.

The decision to create two different types of spectrograms (mel spectrogram for the entire event and mel spectrogram for the frame) is due to the need to determine through testing which of the two solutions is the best with regards to the inference set available. On one hand, creating a spectrogram per frame, without leveraging the frame division inherent in the mel_spectrogram function of librosa, loses the context entirely, especially with 10 ms frames. However, it makes it much easier to classify short-duration events in task as the bio-acoustic one in this case. On the other hand, creating spectrograms from longer audio segments, as large as the event itself for the positive support set, makes the spectrogram effectively complete with context and all the information of the event, allowing for better pattern recognition, although it is less good at handling very short audio segments.

Naturally even if the mel spectrogram is created based on an event of x sec, the mel function will divide anyway the resulting mel in frames of the same duration of 10ms. The difference is only the way the mel spectrogram is created.

From an implementional point of view the functions which create the support and query sets are quite similar. Both uses the label structure and the support set dictionary created before. Below will be reported a pseudocode graph visualization for the algorithm:



Algorithm 2 Support and Query Set creation for each episode

Data: Audio files, class labels

Result: Save episodes to file

```
foreach audio do
    foreach class in audio do
        support_events = GetSupportEvents(audio, class) support_spectrograms = []
        foreach event in support_events do
            spectrogram = CreateSpectrogram(event)
            foreach frame in spectrogram do
                tuple = (frame, class_label) support_spectrograms.append(tuple)
        segments = GetSegmentsBetweenEvents(audio, class)
        foreach segment in segments do
            spectrogram = CreateSpectrogram(segment)
            foreach frame in spectrogram do
                tuple = (frame, negative_label)
        query_fragments = GetQueryFragments(events) query_spectrograms = []
        foreach fragment in query_fragments do
            spectrogram = CreateSpectrogram(fragment)
            foreach frame in spectrogram do
                tuple = (frame, frame_label) query_spectrograms.append(tuple)
    episode = (class_label, support_spectrograms, query_spectrograms) SaveToFile(episode, file_path)
```

For the raw audio the code creation phase is way easier, therefore here won't be reported the pseudo code. The key points of the raw data support set and query set creator algorithm are the same of the aforementioned spectrogram creator algorithm. The differences are the missing necessity of making the spectrograms, therefore the audios are just loaded with the librosa library and then divided into frames and labelled using the list of tuples structure. Considering the specification of Wav2Vec, the model which the raw audio algorithm creates datas for, it has been necessary to sample the audio using a sample rate of 16Khz. Differently from the 22.05KHz used until now.

In this regard, it has been necessary to create another support and query creation algorithm specifical for AST. This new algorithm has been necessary because AudioSet (the dataset used to pre-train AST) is sampled at 16kHz. Therefore, it has been necessary to create a final piece of code that allows obtaining the data in a way not dissimilar to the algorithms already discussed, but with audio sampled at 16kHz

3.3. Data Augmentation

The few-shots learning nature of the task as said before causes a huge data scarcity problem. Mitigate this dangerous situation is a key point that must be done in order to have a robust model capable of achieving good results. To do so the most obvious and at the same time the most effective solution is Data Augmentation (also referred as DA). Data Augmentation is the process of creating new datas starting from the already available ones. In a complex scenario as this one, every data created must follow a golden rule, the sample generated must be credible [23]. For example, in a human speech recognition scenario, if the data augmentation operations involve pitch shifting, the frequency shift can't put the audio in a frequency band where no human sounds can be involved.

Data augmentation if done in a correct way can improve not only the lack of datas but even increase the model robustness to noises and even the accuracy of the model. Moreover, it can lead to a substantial reduction of overfitting.

DA in an audio domain is inevitably different in modalities from computer vision data augmentation. In computer vision there are lot of transformations like crop, scaling, rotate or flipping. When working with audio data these operations can't be used since the format used are raw audio or spectrograms.

3.3.1 Data augmentation for the support set

In regards to this specific scenario the data scarcity comes from the abysmal amounts of samples (k=5 events available). Therefore, the data augmentation is done on these support samples in 2 different modalities: data augmentation for the raw audio corresponding to the events and data augmentation for the spectrograms corresponding to the events and the already augmented frames.

For raw audio the Data Augmentation transformation used in this work are:

- Time shifting: As the name suggests it makes the audio waveform shift in a circular way. This operation improves the model robustness to advances or delays therefore, simulating real situations where signals can be out of synchronization.
- Pitch shifting: Change the tonality of an audio signal without changing the audio signal speed. Pitch shifting can help simulating different speech situation and give the model robustness to shift changes common in real situations.
- White Noise addition: Operation which adds white noise randomly to the raw audio signal. Helps the model being more robust to interferences and background noises.
- Time stretching: Tecnique which simulates a speed increase/decrease of the audio, therefore changing the audio signal duration. Thanks to time stretching is possible to help the model handle changes of speed execution for events in inference.
- Reverb: Reverb is a data augmentation technique which simulates the effect of sound reflecting off surfaces in an environment, creating an echo or "tail" to the sound. This effect can be used to improve the robustness of machine learning models to different acoustic conditions and to make training data more realistic.
- Medium-high frequencies emphasis: Using a High-Pass filter it’s possible to allow frequencies above a certain threshold to pass through the filter while the frequencies below are attenuated.
- Filtering: At the same time using a low-pass filter only the low frequencies are allowed. If the filter used is the Band-Pass one is possible to let frequencies in a specific band to pass through, attenuating the others outside of the band. [24]

For spectrograms the Data Augmentation transformation used in this work are:

- Time masking: Operation which involves randomly selecting a block of consecutive time steps in the spectrogram and setting them to zero. It is possible then to simulate scenarious where parts of the audio are missing or are occluded, helping the model to focus on the overall patterns in the data rather than relying too heavily on specific time segments.

- Frequency masking: Frequency masking is done by randomly selecting a block of consecutive frequency bins in the spectrogram and setting them to zero. This operation simulates situations where frequency component are missing, encouraging the model to rely on other parts of the spectrum and overall patterns rather than specific frequency ranges.

The primary objective of all these different operations is to increase at least 20 times fold the dimension of the support set. Codewise, to apply the data augmentation are used 2 functions, augment_audio and augment_spectrograms which apply a random combination of the transformations discussed before. Using these functions, the support set dimension can be multiplied by the 20 times folds needed for the raw audio. Therefore, making the data scarcity caused by the few-shots scenario less penalizing.

The data augmentation has been made to support the creation of robust prototypes. This in conjunction with a dedicated training phase of the prototypical network to do good prototypes, allow the model to achieve a robust representation for the support class for each given episode. As said before few-shots learning scenarious need a model capable of being good not only to do a classification/localization but also generalize with literally 5 events. Therefore, the need of handling the problem as an episodic meta-learning. The model must learn even how to learn in the most optimized way.

This data augmentation format allows the application of the transformations both to individual frames and, especially, to events, enabling the creation of both synthetic frames and events.

3.4. Models specifications

This section will discuss the three different models used in this research. These models are used with two different but necessary application: first as an encoder to compute the embeddings given a frame/event and second for multiclass classification to improve the learning process. The different model types are capable of creating robust representations, each in a unique way based on their intrinsic specifications.

3.4.1 Convolutional Neural Network Model

The first and most meticulously developed model used in this project is the CNN model. The architecture consists of an encoder with three 2D convolutional layers, each followed by batch normalization, ReLU activation, and max pooling, effectively capturing spatial patterns from 2D inputs like spectrograms.

The flattened output from the final convolutional layer is passed through a fully connected layer that reduces the dimensionality to a hidden representation of dimensionality 768. The optimal hidden dimension that balances performance and results was found using an empirical approach to be nearly 750. Therefore, has been decided to use a hidden dimensionality of 768, the same as the one used in Wav2Vec. This hidden representation is then fed into a multiclassification head used for a multiclass classification auxiliary task. From a design perspective, the particularity of the model is the necessity of making a MaxPooling only on the first dimension, caused by the dimensionality of the input spectrogram frame of (128,2). The value 2 in the second dimension corresponds to the number of samples contained in a 10ms temporal frame using a sample rate of 22KHz.

The CNN encoder is not pre trained on any external dataset. This has been decided to make a comparison with the others pre-trained model. Moreover, it has been deemed necessary to design a model which can achieve good performance even with only the few shots intrinsic in the task.

3.4.2 Audio Spectrogram Transformer Model

The second model used in this work is an Audio Spectrogram Transformer (AST) one, which has been specifically fine-tuned for audio processing tasks on AudioSet, integrating state of the art transformer-based techniques with additional neural network layers for specialized classification. The core component of the model is a pre-trained AST encoder, obtained from the MIT repository, which is designed to extract sophisticated audio features from spectrogram inputs. This encoder exploits self-attention mechanisms, implemented through scaled dot-product attention, to effectively capture temporal and spectral dependencies in the audio data, if present.

To maintain the integrity of the pre-trained representations, all parameters of the AST encoder are frozen during training, ensuring that the model retains the high-level abstractions learned from the large-scale audio dataset it has been trained on. Following the encoder, the model incorporates two intermediate fully connected layers: the first expands the feature dimensions by a factor of four, enhancing the model's capacity to capture intricate patterns, while the second reduces the dimensionality back to the original hidden size, refining the features for subsequent classification. This expand-reduce structure of the intermediate layers has been choosen to be similar to how AST is designed to work in the encoder. These layers are necessary to make the model able to improve in making the embedding representations. These representations will have a pivotal role in classifying each frame of an audio using the positive and negative prototypes as a comparison.

The refined embeddings are then processed by an auxiliary classification head which has two linear layers, designed for multiclass classification tasks, allowing the model to perform a broader range of predictions based on the learned features. This head as discussed for the CNN model is not responsible of making the classification and localization in inference phase, but it is used as an auxiliary head for learning proposal during training. The overall design of the AST model, with its combination of a powerful transformer-based encoder and additional fully connected layers, provides a robust framework for tackling complex audio classification tasks with state of the art performances.

3.4.3 Wav2Vec Model

The third and last model used for this complex task is a custom Wav2Vec2 Model designed for specifically exploiting the Wav2Vec2 excellent architecture to perform feature extraction and classification for the bio-acoustic event classification task. The model integrates a pre-trained Wav2Vec2 encoder, which as already discussed is known for its ability to learn rich audio representations from raw waveform inputs through self-supervised learning. This encoder is equipped with advanced attention mechanisms, in this case, "flash attention 2.0," to efficiently capture and process temporal dependencies in the audio data.

As already seen for the AST model, also the Wav2Vec encoder is already pre-trained. In this case the pre-trained model chosen for the task is the large version pre trained on 960 hours of sampled speech audio (wav2vec2-large-960h-lv60). Therefore, the encoder's parameters are frozen to preserve the valuable features learned during pre-training, ensuring that the model retains the generalization capabilities developed from the large-scale audio dataset.

To adapt the extracted features for specific downstream tasks, the model introduces a single intermediate fully connected layer that reduces the dimensionality of the embeddings produced by the Wav2Vec2 encoder. This layer transforms the high-dimensional output of the encoder into a more compact representation, better suited for the subsequent classification tasks.

Lastly the model architecture is composed by a multiclass classification head. This head is implemented as a linear layer that maps the transformed embeddings to a predefined number of output classes (in this case, 22). This multiclass classification head is responsible for distinguishing between multiple categories during training, making the model versatile for various audio classification challenges. The overall architecture, combining the powerful Wav2Vec2 encoder with a task-specific classification head, provides a robust framework for handling complex audio data, allowing for accurate and efficient event prediction.

3.5. Tasks suddivision

The complexity of the task, combined with the three different and competitive types of encoders, necessitates a carefully designed and intricate task handling and breakdown. As already discussed during the data creation discussion, it has been decided to divide the training portion of the task in 54 subcases, referred to as episodes. This is the reason why all data operations so far have aimed at creating 54 files, each representing a training episode.

Using the prototypical network with one between AST, CNN or Wav2Vec encoder, it has been possible to include a meta-learning approach that allowed learning not just the specific features of the 22 classes present in training, but rather how to effectively create a robust representation of a certain class, regardless of which one it is.

To maximize the meta learning approach of the task it has been chosen to approach the training phase in the same exact way the model would make the inference. Therefore, during the training phase the objective of the model will be to classifiy and localize positive events for a specific class, using only the k=5 support event for each episode. To address such a complex task, it was necessary to divide the macro-task into 2 sub-tasks:

- The main task, based on the generation of prototypes for the support positive events and negative support events. Once successfully created, the positive and negative prototypes, centroid of all embeddings for each type, has been used with a Triplet Loss function to train the model to progressively improve the quality of its embeddings.
- The auxiliary task, based on a multiclass classification task, has been fundamental to increase the overall performances and capabilities of the model, by enhancing the quality of the embeddings learned.

The choice of the use of the triplet loss for the main task has been born from the necessity to maximize t the quality of the embedding space, where "quality" refers to creating representations that minimize the distance between embeddings of similar classes and maximize the distance between embeddings of different classes. Another loss function which would have been perfect for this task would have been the contrastive loss, however the triplet loss has been deemed better in this situation with refer to the possibility of exploiting the negative support set. In fact, in the support set there is a huge number of negative examples. These abundant negative frames can be used by the triplet loss to optimize not only the distance between positive events's embedding but also to increase the distances by the use of the margin which ensures that the anchor is closer to the positive example than to the negative example by a certain margin. The triplet loss has a structural definition of: TripletLoss(Anchor,Embedding_with_the_same_class_as_the_anchor,Embedding _of _the_other_class). This in conjunction with the binary classification nature of the real task, permits the model to learn not only how to make effective embedding but also to distanziate them from the surrounding audio portions.

The choice of using a multi-class classification task as an auxiliary one for the whole model has been made based on the practical need for the model to improve generalization and enhance the feature learning. By learning to classify into multiple categories, the model develops embeddings that better capture the underlying structure of the data. Moreover, it can improve the stability of training by providing additional supervision and reducing the risk of the model becoming too focused on a single objective.

3.6. Training Procedures

The implementation phase of the main and the auxiliary task in the training loop has been made ad hoc for each encoder. In this paragraph will be discussed firstly the common part that unites the three different implementations and then once at a time will be eviscerated the specific part of the training loop for each of the models used.

3.6.1 Shared elements of the algorithms

The algorithms are created with regards to the need of managing the support set, which consists of k=5 positive events, the synthetic positive events created through data augmentation of the positive events and the negative support set portions interspersed between the positive events. The algorithm first processes a loop that allows it to iterate through the support set using batches called “support set loop”.

For the “support set loop” has been defined two different modalities to compute the prototype for the support positive and support negative. The first method is what’s called rolling mean: instead of collecting all the embeddings for a class, and compute the mean all at once, it’s possible to make the mean of the first two with a third one and so on. This mathematical approach gives naturally more importance to the last embedding computed, but at the same time permits a better use of the memory in devices with prestational or energetic problems. The second mean computation method is the classical centroid mean between all the embeddings. Both approaches still allow for obtaining two embeddings in the end: one positive, which we will call the positive prototype, and one negative, referred to as the negative prototype.

The second sequential loop is the one which iterates the query set using batches, as seen in the support set loop. The query set created in the data processing phase is in the format (spectrogram/raw_waveform, label_for_the frame). However, to successfully execute the training algorithm, it is necessary to have labels not only in a multiclass format but also in a binary format with only labels 0 and 1. Here, 0 corresponds to all classes, including the class labeled as 0 in the multiclass format (Unknown class) minus the class which must be classified and localized for the episode. This dual approach will allow for simultaneous handling of both the auxiliary multiclass classification task and the primary binary classification task for the entire query set. Therefore, in the "query set loop," three main phases can actually be distinguished:

- Computation of the embeddings and the logits, using the multiclass classification head, for the batch.
- Computation of the main task loss using the triplet loss function. Based on the current label of the batch, the triplet loss will be computed in a different way, using as the second argument of the function the prototype of the same class as the one computed on the query batch.
- Computation of the auxiliary task loss using a classic cross entropy loss function. This function gives a greater loss penalty if the model predicts a class with a high probability and that prediction is wrong. Therefore, making it a great addition as an auxiliary loss for the microtask. Thanks to this secondary task the model can learn how to differenciate different classes other than the 0 class and the positive one for the episode, allowing the model to learn a more robust representation.

At the end of these phases, a multi-loss is calculated by combining the previously computed binary triplet loss and the auxiliary task loss. A lower coefficient (0.8) is assigned to the auxiliary task loss to prevent the model from overfocusing on classifying the 22 classes during the training phase rather than becoming excellent in creating the prototypes. This balance ensures that the model concentrates more on improving the quality of the embedding itself, enhancing its ability to generalize and refine the embedding space rather than solely optimizing for class-specific classification.

A key component of the training algorithm is the ability to handle the significant class imbalance in the query set through a parametric setting. Two distinct solutions are proposed to address this challenge: the first involves applying the Synthetic

Minority Over-sampling Technique (SMOTE), which generates synthetic samples for the minority class to balance the dataset. Alternatively, the second approach is to perform an undersampling of the majority class (negative frames/events), ensuring the query set becomes balanced and preventing the model from exploiting the imbalance by consistently predicting the majority class (class 0). It has been decided to primarily use the undersampling technique due to the overwhelming dominance of the majority class (negative frames/events) in the query set which did not permit SMOTE to produce realistic and diverse examples for the minority class, leading to much worse result and a total overfitting of the model.

Despite the presence of a similar imbalance in the support set, this imbalance is not detrimental to the model. In fact, it can be instead beneficial, since the support set is only used to create robust prototypes and is not directly involved in classification. This allows the model to learn more distinctive and reliable negative class representations without biasing the decision-making process.

Moreover, during the query set loop, the three models process the distance between query embeddings and the prototypes using the Euclidean distance function (F.pairwise_distance). These distances, once computed, can be compared to predict if the analyzed embedding is closer to positive or negative and therefore, classifying that frame/event. Doing this for the whole query set, it has been possible to have a numpy array of the same dimension as the query labels containing all the predictions. Confronting these 2 vectors can lead to the computation of significant metrics such as F1 score, Precision and Recall.

To handle potential issues external to the code, it has been decided to save the model parameters and update them at each training episode, ensuring safety and the ability to interrupt the training process.

3.6.2 Training Algorithm for CNN encoder

The algorithm used with the CNN encoder differs from others primarily in its frame-by-frame handling of the support and query sets. Therefore, within the two main loops, embeddings are created for each individual frame. In fact, if the batch size is 32, there will be an inner loop executed 32 times, generating 32 embeddings for each of the frames in the batch. Each of the embedding will then be used for the prototypes creation during the support set loop or for the loss calculation in the query set loop. The Frame-by-frame analysis of the algorithm allows to obtain a very precise level of temporal detail. This is crucial for identifying and classifying specific events that occur at exact moments and with a low temporal size. Moreover, the use of a frame system facilitates the handling of labels by allowing each frame to contain almost exclusively data from a single event. This approach helps avoid the need for voting systems to manage hypothetical events that contain both event fragments and empty fragments.

On the other hand must be said that frame-based management does not allow the convolutional model to fully utilize all the sequential aspects of a bioacoustic event. Indeed, a hypothetical animal call has a beginning and an end, and this cannot be leveraged using this algorithm.

3.6.3 Training Algorithm for AST encoder

To manage the use of AST as a model, it has been necessary to drastically change how batches are handled. Indeed, to have an input suitable for AST, we had to shift from a query and support set in the format (x,128,2) to a format as ($\frac{x}{512}$,128,1024). The shape of these input represents respectively the number of frames/batch size in the training loop, the number of mels bins and lastly the number of time frames. This is necessary to allow the AST pre-trained with AudioSet to successfully compute embeddings. Consequently, it was unavoidable to use batches of size 512 and concatenate all samples in a batch along the third dimension. This also required managing labeling through a voting system, which effectively makes all labels in the batch the same as the most common one. While this allows AST to successfully utilize the attention mechanism, it also skews the data, making the model believe that a batch contains only positive events when, in reality, it contains a small number of negative events as well. As a result, an inevitable degradation in embeddings occurred, even if balanced by AST's actual ability to leverage event sequencing.

During the creation phase of the prototypes, it has been deemed interesting and functional to order the samples based on the label. This is done to maximize the use of the attention mechanism and to minimize the performance degradation introduced by the voting labelling system. In this way the support set will contain only an event with both negative and positive frames, thus reducing the prototypes representation error to the least possible.

3.6.4 Training Algorithm for Wav2Vec encoder

A good balance between the frame system used with the CNN encoder and the event system used with AST is the algorithm used for Wav2Vec. In fact, by using the Wav2Vec processor (which includes the tokenizer and feature extractor), it is possible to successfully obtain representations by transforming from a raw audio form of (x,160) with x representing the batch to a form of (x*160,). Thus, by using a batch size of 32, similar to what is done with the CNN model, it is possible to effectively leverage the Wav2Vec pretrained model with 960 hours of training data.

The output of Wav2Vec naturally adds a central dimension that represents segments of the input that have been transformed called patches. By using this functionality of Wav2Vec, it is possible to manage labels through an algorithm that allows each segment to be labeled based on the two labels that actually constitute that audio fragment. This approach enables the successful use of Wav2Vec while minimizing the penalty associated with managing aggregated labels containing multiple different events.

3.7. Inference Procedure

Due to the training phase being managed in the same way as the actual inference on the external audio, once the model is trained, the inference function does not support any particular novelties. In fact, it operates similarly to the training loop, with the obvious but substantial difference being the absence of a loss function and thus no backpropagation and learning. With this obvious but necessary consideration in mind, the inference algorithm retains from the training procedures the entire support set loop that leads to the creation of prototypes for both the negative and positive classes. In the query set loop, as during the training phase, embeddings are calculated for all frames/events in the query set and compared with positive and negative embeddings. If the embedding is closer to the positive one than the negative, a positive classification is made; otherwise, a negative classification is performed.

To make the code universal and effectively usable, it was deemed necessary to create a new implementation for the section of code related to the creation of the .pth file, which, in training, represented an episode but now refers to the inference of an audio. Therefore, instead of initially preparing data in a suitable format for a multitude of audio files to be fed into the model, it was decided to create a dedicated functions that creates the support and the query set specific to that audio, given a path to a .wav file and a CSV/JSON file containing the event list. Therefore, using the datas from the support set can be created the prototypes for the positive and negative events for that audio and then perform the inference using the query set. This approach allows for the creation of data and inference to be carried out in a single execution, greatly speeding up the overall inference management process.

Once the entire query set is classified, the actual query labels are compared with the model's predictions. This comparison is done using the F1 score. The F1 score is a metric used to evaluate the performance of classification models in scenarios where the classes are imbalanced. In fact, for tasks like those in the DCASE challenge, one could train a model to always predict negative, thus achieving an excellent accuracy rate. However, using the F1 score as a metric ensures that the model's true capability is represented, as it balances precision and recall and cannot be easily circumvented by simply predicting one class. This way, the F1 score provides a more accurate measure of the model's performance in tasks with imbalanced classes, reflecting its effectiveness in distinguishing between positive and negative events. The F1 score, from a theoretical standpoint, represents the harmonic mean of precision and recall, providing a single score that balances both metrics. It has a mathematical definition of:

$$F1\ Score = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

3.7.1 Event and Localization Management

In addition to the classification section of the task with the classic F1 score as a metric principal metric, the second and more complex part of the task is the localization one. Localizing with refer to the literature and the challenge specification, means predicting the start and end of various events for the entire query set. Naturally, it is necessary to establish a certain degree of precision within which to compare the predicted events with the actual events.

To localize the start and end time for events has been necessary to implement 2 fundamental functions:

- The first, called wrong prediction corrector is a function that for each frame prediction verifies if the previous and the following n frame are predicted as the same class. If this condition is not verified, then the prediction for that frame is with a high percentage degree wrong, therefore it will be changed. This procedure has been necessary to comply two different functionalities at once:
    1. Increase overall classification performances. Using an algorithm with n based on the mean event duration in frames, the F1 score discussed before can be increased to new heights.

2. Remove the eventual predicted Pos/Neg frames between streaks of Neg/Pos. In fact, the model can make wrong prediction but using the event base nature of the task is possible to correct the errors. Furthermore, making it possible to localize successfully even more events, which would have not been localized if there was a single negative frame between hundreds of positive ones.

- The second, called event finder, as the name suggests is an effective function which given a vector of predicted or true labels (the ones used to compute classic F1 score) return tuples of start and end of events. These two list of tuples representing events predicted and true events will be then confronted to verify the model capability of localizing events other than frames.

Once the list of tuples containing predicted and true events for the query set has been computed, the next step is to confront them. To do this, it has been decided to use the IoU, an important metric used for segmentation that allows the calculation of the overlap between two sets: the predicted set (the classification output of the model) and the ground truth set (the actual correct segmentation). The formula is:

$$IoU = \frac{Area(\text{prediction} \cap \text{ground truth})}{Area(\text{prediction} \cup \text{ground truth})}$$

Using the IoU metric each event of the predicted event set is confronted with each event of the true event set. If there is a matching between events of at least 95% then the event is successfully classified and not considered anymore for the confront with the true event set. This approach permits to compute directly the true positives, the false positive and the false negatives, which will be used consequently to compute the event based F1 score, Precision and Recall.

4. Chapter 4

4.1. Results

4.1.1 Inference audios analysis

Once the various models were created, the data managed, and the metrics chosen, the next step was to perform inference on the test audio files. With this regard, the test audio files can be divided into four main categories:

The first category includes the audio files called file_x_y, where x and y are two 3 digit number. This category of files contains events with very variable average durations, ranging from a minimum of 2 seconds to a maximum of about 20 seconds. These characteristics make such audio files slightly more complex to classify and localize due to the variability of the events.

Next, there are the audio files called *ME*. This category contains events with little variation in duration, but they are very short, lasting around 0.2 seconds. The short duration of the events makes this category particularly challenging to classify and localize.

The third category of audio files includes those that start with *BUK*. These audio files are extremely difficult to handle, as they contain events with durations ranging from 0.03 to 0.1 seconds. The extreme variability combined with the short event durations makes these audio files the most challenging, and the results reflect this difficulty.

Last but not least, there are the audio files that start with *R4*. These audio files have events with durations ranging from 6 to 15 seconds, making classification and localization somewhat complex but significantly more manageable compared to the categories discussed so far.

4.1.2 Mean results

To summarize the results before commenting on and analyzing them, it can be stated with certainty that the models performed in a fairly similar manner. In fact, considering the convolutional model with spectrograms created on a single frame and the one with spectrograms created on the average event as two distinct models, have been found a situation where the worst-performing model based on the localization f1-score is the convolutional model that handles the creation of spectrograms frame by frame. This model produced the following data:

Class. F1 score	**Loc. F1 score**	**Precision**	**Recall**
37,12%	21,1%	21,03%	31,06%

Subsequently, with extremely similar performance, we have Wav2Vec, which achieved the following results:

Class. F1 score	**Loc. F1 score**	**Precision**	**Recall**
35,54%	21,13%	29,29%	49,3%

The second-best performing model is AST, which achieved the following metric values during inference:

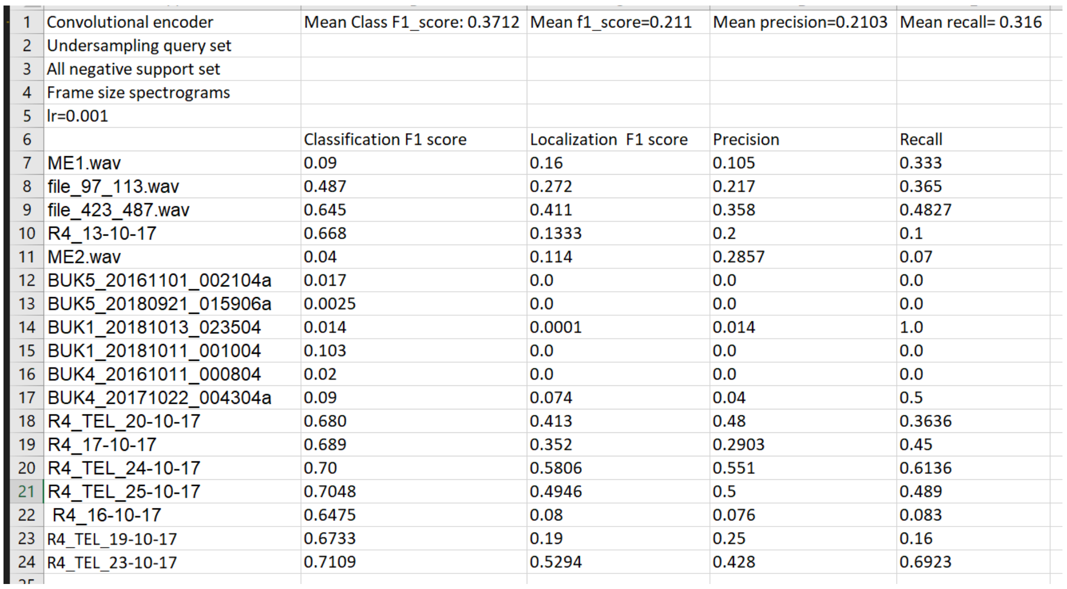
Class. F1 score	**Loc. F1 score**	**Precision**	**Recall**
35,37%	25,10%	45,12%	24,19%

The model that achieved the best performance is, in fact, the convolutional model with spectrograms managed through the average event duration. Practically, it achieved the following milestones:

Class. F1 score	**Loc. F1 score**	**Precision**	**Recall**
37,7%	25,75%	28,70%	34,49%

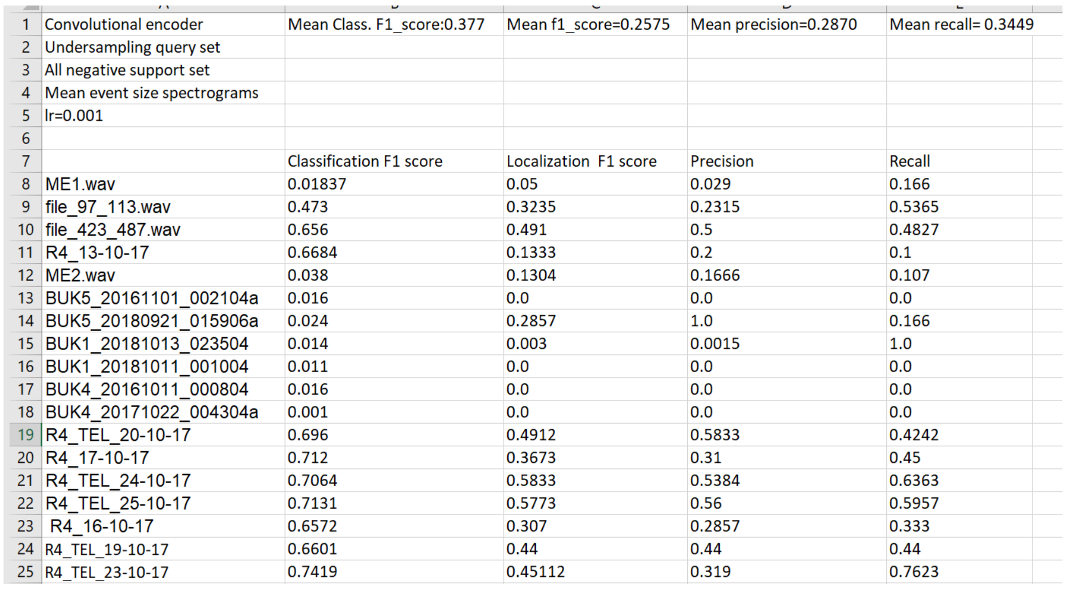
In the following sections, various charts will be presented, showing all the most important statistics for each audio. There will be one chart per model.

4.1.3 CNN with frame spectrogram



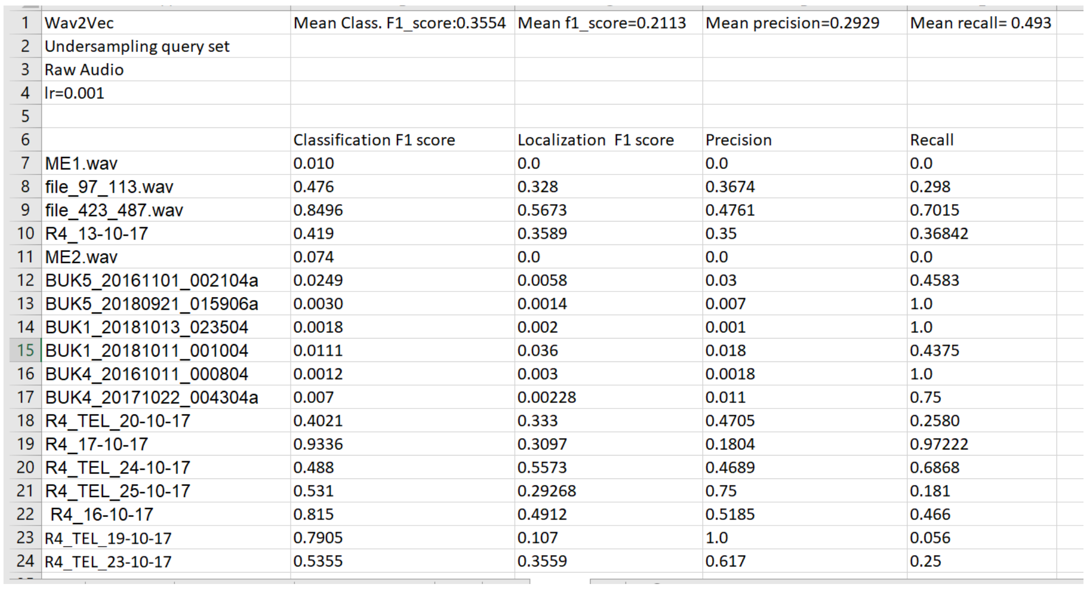
Convolutional encoder	Mean Class F1_score: 0.3712	Mean f1_score=0.211	Mean precision=0.2103	Mean recall= 0.316
Undersampling query set				
All negative support set				
Frame size spectrograms				
lr=0.001				
	Classification F1 score	Localization F1 score	Precision	Recall
ME1.wav	0.09	0.16	0.105	0.333
file_97_113.wav	0.487	0.272	0.217	0.365
file_423_487.wav	0.645	0.411	0.358	0.4827
R4_13-10-17	0.668	0.1333	0.2	0.1
ME2.wav	0.04	0.114	0.2857	0.07
BUK5_20161101_002104a	0.017	0.0	0.0	0.0
BUK5_20180921_015906a	0.0025	0.0	0.0	0.0
BUK1_20181013_023504	0.014	0.0001	0.014	1.0
BUK1_20181011_001004	0.103	0.0	0.0	0.0
BUK4_20161011_000804	0.02	0.0	0.0	0.0
BUK4_20171022_004304a	0.09	0.074	0.04	0.5
R4_TEL_20-10-17	0.680	0.413	0.48	0.3636
R4_17-10-17	0.689	0.352	0.2903	0.45
R4_TEL_24-10-17	0.70	0.5806	0.551	0.6136
R4_TEL_25-10-17	0.7048	0.4946	0.5	0.489
R4_16-10-17	0.6475	0.08	0.076	0.083
R4_TEL_19-10-17	0.6733	0.19	0.25	0.16
R4_TEL_23-10-17	0.7109	0.5294	0.428	0.6923

4.1.4 CNN with event spectrogram



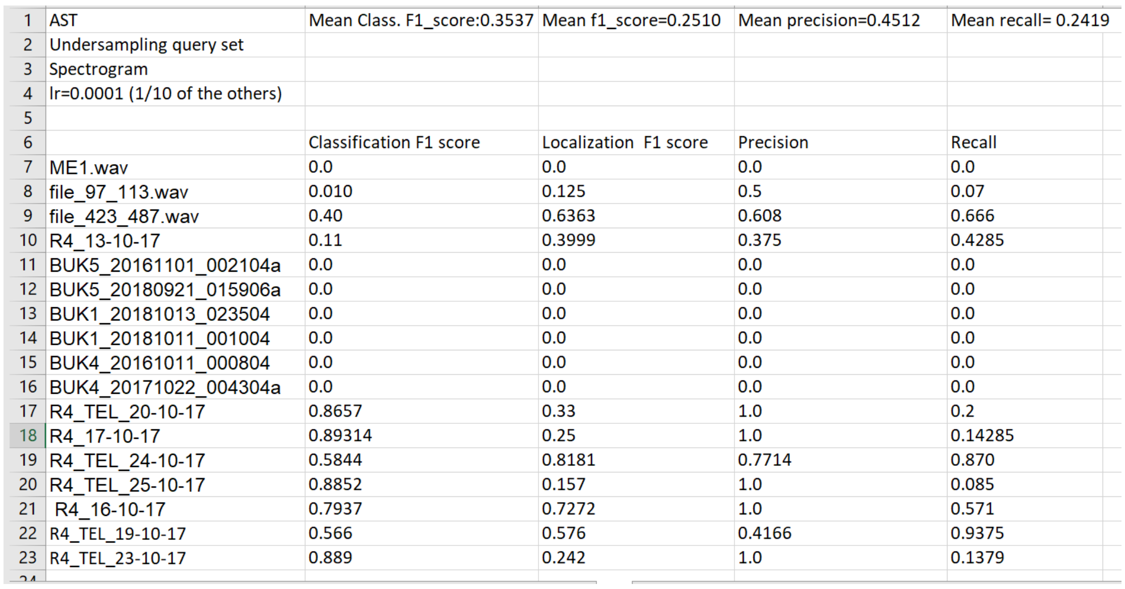
1	Convolutional encoder	Mean Class. F1_score:0.377	Mean f1_score=0.2575	Mean precision=0.2870	Mean recall= 0.3449
2	Undersampling query set				
3	All negative support set				
4	Mean event size spectrograms				
5	lr=0.001				
6					
7		Classification F1 score	Localization F1 score	Precision	Recall
8	ME1.wav	0.01837	0.05	0.029	0.166
9	file_97_113.wav	0.473	0.3235	0.2315	0.5365
10	file_423_487.wav	0.656	0.491	0.5	0.4827
11	R4_13-10-17	0.6684	0.1333	0.2	0.1
12	ME2.wav	0.038	0.1304	0.1666	0.107
13	BUK5_20161101_002104a	0.016	0.0	0.0	0.0
14	BUK5_20180921_015906a	0.024	0.2857	1.0	0.166
15	BUK1_20181013_023504	0.014	0.003	0.0015	1.0
16	BUK1_20181011_001004	0.011	0.0	0.0	0.0
17	BUK4_20161011_000804	0.016	0.0	0.0	0.0
18	BUK4_20171022_004304a	0.001	0.0	0.0	0.0
19	R4_TEL_20-10-17	0.696	0.4912	0.5833	0.4242
20	R4_17-10-17	0.712	0.3673	0.31	0.45
21	R4_TEL_24-10-17	0.7064	0.5833	0.5384	0.6363
22	R4_TEL_25-10-17	0.7131	0.5773	0.56	0.5957
23	R4_16-10-17	0.6572	0.307	0.2857	0.333
24	R4_TEL_19-10-17	0.6601	0.44	0.44	0.44
25	R4_TEL_23-10-17	0.7419	0.45112	0.319	0.7623

4.1.5 Wav2Vec



1	Wav2Vec	Mean Class. F1_score:0.3554	Mean f1_score=0.2113	Mean precision=0.2929	Mean recall= 0.493
2	Undersampling query set				
3	Raw Audio				
4	lr=0.001				
5					
6		Classification F1 score	Localization F1 score	Precision	Recall
7	ME1.wav	0.010	0.0	0.0	0.0
8	file_97_113.wav	0.476	0.328	0.3674	0.298
9	file_423_487.wav	0.8496	0.5673	0.4761	0.7015
10	R4_13-10-17	0.419	0.3589	0.35	0.36842
11	ME2.wav	0.074	0.0	0.0	0.0
12	BUK5_20161101_002104a	0.0249	0.0058	0.03	0.4583
13	BUK5_20180921_015906a	0.0030	0.0014	0.007	1.0
14	BUK1_20181013_023504	0.0018	0.002	0.001	1.0
15	BUK1_20181011_001004	0.0111	0.036	0.018	0.4375
16	BUK4_20161011_000804	0.0012	0.003	0.0018	1.0
17	BUK4_20171022_004304a	0.007	0.00228	0.011	0.75
18	R4_TEL_20-10-17	0.4021	0.333	0.4705	0.2580
19	R4_17-10-17	0.9336	0.3097	0.1804	0.97222
20	R4_TEL_24-10-17	0.488	0.5573	0.4689	0.6868
21	R4_TEL_25-10-17	0.531	0.29268	0.75	0.181
22	R4_16-10-17	0.815	0.4912	0.5185	0.466
23	R4_TEL_19-10-17	0.7905	0.107	1.0	0.056
24	R4_TEL_23-10-17	0.5355	0.3559	0.617	0.25

4.1.6 AST



1	AST	Mean Class. F1_score:0.3537	Mean f1_score=0.2510	Mean precision=0.4512	Mean recall= 0.2419
2	Undersampling query set				
3	Spectrogram				
4	lr=0.0001 (1/10 of the others)				
5					
6		Classification F1 score	Localization F1 score	Precision	Recall
7	ME1.wav	0.0	0.0	0.0	0.0
8	file_97_113.wav	0.010	0.125	0.5	0.07
9	file_423_487.wav	0.40	0.6363	0.608	0.666
10	R4_13-10-17	0.11	0.3999	0.375	0.4285
11	BUK5_20161101_002104a	0.0	0.0	0.0	0.0
12	BUK5_20180921_015906a	0.0	0.0	0.0	0.0
13	BUK1_20181013_023504	0.0	0.0	0.0	0.0
14	BUK1_20181011_001004	0.0	0.0	0.0	0.0
15	BUK4_20161011_000804	0.0	0.0	0.0	0.0
16	BUK4_20171022_004304a	0.0	0.0	0.0	0.0
17	R4_TEL_20-10-17	0.8657	0.33	1.0	0.2
18	R4_17-10-17	0.89314	0.25	1.0	0.14285
19	R4_TEL_24-10-17	0.5844	0.8181	0.7714	0.870
20	R4_TEL_25-10-17	0.8852	0.157	1.0	0.085
21	R4_16-10-17	0.7937	0.7272	1.0	0.571
22	R4_TEL_19-10-17	0.566	0.576	0.4166	0.9375
23	R4_TEL_23-10-17	0.889	0.242	1.0	0.1379

4.2. Result discussion

Once all the results from the developed models have been presented, it is now necessary to discuss their performance and the characteristics that make them unique. However, before starting, it is essential to state that the results obtained are much lower than they would have been without the BUK audio class. In these audio files, all four presented models achieved null or nearly null F1 score performances, failing to derive a sufficient representation. As a result, none of the models were able to achieve significant results for this audio class, without localizing events of such short duration from the few dozen frames of 10 ms given as the support set.

If not considered the best model presented until now would have had a mean f1 score for classification of 60.56% and an f1 score for localization of 38%. The AST model in this same scenario would have obtained a mean classification f1 score of 49% and a mean localization score of 38.74%.

Having made this necessary preamble, by analyzing the results of nearly all the models, we can confidently state that good results were achieved, even considering the worst audio, in line with the 2023 challenge baseline. In this regard, the critique that arises from reviewing these results is inevitably the need to improve event management. Indeed, while each model achieved excellent peaks in the classification of individual frames, these peaks considerably drop when considering the F1 score calculated for localization.

Going into more detail on the results of the various models, an extremely interesting aspect that merits analysis is the fact that the CNN and Wav2Vec models have a recall much higher than precision. This means that these models predicted many more positive events than there actually were, resulting in a significant number of false positives.

In contrast, AST achieved a much lower recall, classifying fewer positives compared to the other three models. However, this model has a markedly higher precision, which effectively reduces the false positives to a historic minimum between the models. Therefore, AST in this case is a more suitable model in the cases where the designer needs less positive classification and localization but more accurate.

At the same time if the scenario requires an event localizer where it doesn’t matter which animal is found, the high recall of models like CNN and Wav2Vec might be advantageous because they ensure that most potential events are detected, even if it means including some false positives. This approach is useful when capturing as many events as possible is more important than precision. This trade-off can be crucial depending on the specific requirements of the application.

Finally, an important aspect to highlight is that Wav2Vec, compared to the other models, performed below expectations. This is likely due to the dataset on which it was trained. Although Wav2Vec is an incredible model in terms of architecture, it is trained primarily on human speech audio. This pre-training on speech data results in the model not only failing to achieve better results but even degrading its performance when dealing with bioacoustic events such as animal calls.

On the other hand, AST, despite also being pre-trained, performs excellently due to the universality of the AudioSet dataset on which it was pre-trained. Therefore, using such a dataset for pre-training can be very beneficial for creating an AST model aimed at classifying and localizing bioacoustic events.

4.3. Approach's limitations

The approach taken in this thesis, while innovative in applying few-shot learning to bio-acoustic event classification and localization, has several limitations that must be acknowledged.

First of all, the use of only classic data augmentation without innovative and complex techniques in conjuction with the small size and lack of diversity in the dataset represent one of the primary limitations. Few-shot learning inherently suffers from overfitting, and the limited number of the bio-acoustic events avaible makes it challenging to generalize the models to different environments or species not represented in the training data.

Moreover, the complexity of some models, particularly AST and Wav2Vec, requires high computational resources. This poses challenges for deploying these models in real-time or resource constrained environments, such as on edge devices or field-based bio acoustic monitoring systems. Wav2Vec, for instance, performed exceptionally well but relies on pre-trained data, which might not always align with the specific characteristics of bio-acoustic data, leading to suboptimal fine-tuning for specialized applications.

Finally, while the use of prototypical networks showed promise in handling few-shots learning scenarios, their performance is heavily dependent on the quality of the embeddings. Poor embeddings lead to ineffective prototypes, especially in cases where event classes share similar acoustic characteristics. Moreover, as seen in the result section the model to achieve excellent performances requires event long enough to extract successful representations. In scenarious where the event are brief, lasting only a few moments, the model will not be able to obtain a good representation of the positive event to achieve reliable perfomances.

The lack of scalability, especially in dealing with a wider variety of bio-acoustic events, remains as a primary limitation of this approach.

4.4. Lesson Learned and Future Developments

Throughout the development of this project, several interesting ideas and research directions emerged. However, due to time constraints or technical complexity, these proposals were not implemented. In this section will be discussed some of these ideas, outlining their potential and explaining why they were not included in this work.

4.4.1 Lesson Learned

During the research and creation of this work, it was inevitable to question and generate a multitude of ideas. However, as is often the case in both academic and non-academic research, many ideas are not functional. Among the most interesting ideas to mention are:

- Undersampling the negative support set: As seen briefly in the result section, undersampling the negative support set to achieve a balance for the support set has been detrimental to the model performance. From an analytical point of view this result makes totally sense, in fact the unbalancing intrinsic in the dataset is dangerous only with regards to the model classification. Therefore, for the query set in training phase it’s necessary to have a balance, however for the support set means only losing precious datas which would have been fundamental to classify and localize frames and events in the query set.

- Equalizing event durations through data augmentation: This approach is theoretically very interesting. If all events could be made uniform in duration, it could have been entirely resolved the label management issue highlighted during the discussion of training algorithms for AST and Wav2Vec. Alternatively, it could also allow for a more coarse-grained approach, classifying directly the event rather than on a frame-by-frame basis and then apply an event finder function. However, the challenge of this approach lies in finding a functional and effective method to equalize the event duration. In fact, it is not possible to make all events as long as the shortest one due to the extreme variability in their durations and the inevitable information loss. At the same time, making each event as long as the longest one is not a feasible solution, unless a method of augmentation is found that allows the event to maintain its informational significance. Therefore, despite its high potential, the focus of the work was shifted to the frame-based approach, which is central to this research.
- Using an event approach with the CNN encoder: On one hand, handling the task at an event level rather than frame-by-frame in a model such as an RNN or a transformer makes perfect sense and yields interesting results. On the other hand, applying the same approach with a CNN model results in significantly worse performance. This should not be surprising, as managing an event-based approach requires careful handling of the event duration. Even hypothetically finding an effective and elegant solution, the CNN would not benefit from this approach, as it cannot leverage the context or the sequential nature of an event.

4.4.2 Advanced Data augmentation

The use of advanced data augmentation techniques, such as integrating Generative Adversarial Networks (GANs), was considered to increase the dataset's variability. This could have significantly improved the model's ability to generalize to unseen data. In fact, the adversarial approach could have been pivotal to increase the amount of data and therefore managing the data scarcity intrinsic in the few-shots scenario. However, the complexity involved in developing and training GANs in a complex field as the bioacustic one, required setting aside this idea in favor of more traditional and proven data augmentation techniques.

4.4.3 Predictions Feedback System

Another extremely interesting idea that would have been appropriate to develop is the use of what is known in the scientific community as a Feedback system. During the inference phase, using a threshold chosen empirically, it would be possible to use frames/events for which the model is highly confident in its predictions as additional data for the prototypes' creation, leading to a significal improvement of the model's performance. In fact, in a data-scarcity scenario like the task of this research, obtaining additional data with a reasonable degree of certainty allows the model to generalize much better and learn the characteristics of the specific audio more effectively. This technique can be applied not only to the valuable and scarce positive frames but also to the negative ones, creating a chain reaction where the more the model predicts, the better it becomes at making predictions.

4.4.4 Ensemble learning

Last but not least among the interesting ideas not yet implemented is the use of multiple models and in this case, all three models discussed in this research, to perform simultaneous classification and localization of an audio. This approach, also known as ensemble learning, uses a voting system, which can be weighted or not, to exploit the diverse characteristics of different models to achieve a much more effective classification. However, this idea could not be realized because it goes against the rules of the DCASE Challenge 2024, which is a testament to how effective ensemble learning is. From this point of view, it can be said that banning the ensemble learning was the only possible solution to encourage the scientific community to explore other avenues.

5. Conclusions

This thesis started with and was fundamentally aimed at exploring the classification and localization of bioacoustic events under the challenging few-shot condition of Task 4 of the DCASE Challenge 2024. The growing demand for advanced monitoring techniques underscores the necessity for more robust and efficient methods in this domain. This research leverages Few-Shot Learning (FSL) techniques to tackle the problem of limited labeled data, employing a Prototypical Network enhanced by a dual loss function, comprising triplet loss and cross-entropy loss, to improve the model's generalization capabilities from minimal examples.

In the first chapter of this research, all the fundamental concepts for managing a task of this level of complexity were thoroughly examined. In particular, general audio machine learning was discussed along with its inherent peculiarities and complexities, followed by a meticulous analysis of the state of the art in this field. Finally, theoretical concepts of great importance, such as transfer learning and meta-learning, were discussed, with a focus on the prototypical networks, which are central to this work.

During the second chapter, Task 5 of the DCASE Challenge 2024 was explained and discussed. In this context, extensive attention was given not only to the general specifications but also to their implications for the task and its overall complexity. The focus then shifted to the challenge dataset, highlighting the differences between the training and inference datasets. Moreover, some of the most interesting and stimulating ideas from teams that participated in the past two editions were eviscerated.

After discussing the task specifications, the third chapter of this thesis work shifted to examining the actual approach adopted for the task. It focused on how to create and manage a localization system using frames and how to maximize the quality of the representation of a bioacoustic event. The chapter then addressed data management operations: starting from the CSV file provided by the challenge team, various complex functions were used to create the crucial support set, the query set, and the labels in multiple formats. Given that the nature of the task is based on data scarcity, it was essential to perform and process all data augmentation operations to significantly increase the amount of useful data, thus enhancing the performance and effectiveness of the models used.

This work then analyzed with expertise various encoders, including the Audio Spectrogram Transformer (AST), Wav2Vec, and Convolutional Neural Networks (CNNs), to process and analyze bio-acoustic data represented as Mel Spectrograms or Raw Audio Waveforms for the Wav2Vec model. By doing so, it seeks to advance the understanding of how different encoders can be effectively applied to bio-acoustic event detection and classification. Moreover, the research aims to assess the effectiveness of these Few-Shot Learning approaches in improving classification accuracy and to develop precise localization methods for bio-acoustic events within audio recordings.

A focal point of this chapter was the discussion of the complex and extensive training algorithms designed specifically for the three models addressed. These algorithms, based on task subdivision, adeptly manage prototype creation along with the subsequent training phase through a dual loss approach. This approach consists of a primary classification task using prototypes and triplet loss, combined with a secondary multiclass classification task.

Finally, has been examinated thoroughly the management of the inference phase. Structurally, due to the meta nature intrinsic to the task, it was handled in the same way as the training phase. To analyze the statistics of the various models without bias, the F1 score was chosen as the main metric. The F1 score was then presented both in its classic form to evaluate the models' ability to classify the various frames and using a segmentation metric, such as Intersection over Union (IoU), to calculate the F1 score. This approach provides an objective and functional evaluation of the localization quality for the various implemented models.

The final chapter, and indeed the last step of this thesis, discusses the results and proposes future ideas and considerations to expand the scope of this fascinating yet complex task.

Regarding the results, it can be stated with certainty that the presented models met expectations. In the case of the convolutional model, it achieved nearly 61% F1 score for classification and approximately 38% F1 score for localization, not accounting for the BUK class. Moreover, even if the various model presented have achieved similar performances, thanks to a different handling of the event creation it has been possible to have the AST model with high precision in scenario where false positive are detrimental. On the other hand in situation where the specific task needs a low degree of false negative, the CNN model has an high recall, classifying an high percentage of positive event, even with a lower grade of precision.

Therefore, it can be concluded that the models developed and analyzed in this thesis, along with all the techniques introduced, have made a valuable contribution to the scientific community in the field of audio event localization and classification. In particular, this work provides an excellent foundation for starting the study of this complex yet fascinating area of machine learning, offering both practical and theoretical support, along with a robust model that achieves its purpose.

BIBLIOGRAFY

[1]https://towardsdatascience.com/audio-deep-learning-made-simple-part-1-state-of-the-art-techniques-da1d3dff2504

[2]https://waywithwords.net/resource/challenges-in-speech-data-processing/

[3] https://analyticsindiamag.com/topics/representation-learning/

[4].https://www.researchgate.net/publication/261272783/figure/fig4/AS:831704444985344@1575305200100/Visual-representation-of-the-raw-audio-file-for-Western-genre.ppm

[5]https://medium.com/analytics-vidhya/understanding-the-mel-spectrogram-fca2afa2ce53

[6] https://arxiv.org/pdf/2104.01778

[7]https://huggingface.co/docs/transformers/model_doc/audio-spectrogram-transformer

[8] https://arxiv.org/pdf/2006.11477

[9]https://medium.com/@akriti.upadhyay/how-to-make-an-automatic-speech-recognition-system-with-wav2vec-2-0-on-e2es-cloud-gpu-server-f946e1e49196

[10]https://mohitmayank.com/a_lazy_data_science_guide/audio_intelligence/wav2vec2/

[11]https://blog.roboflow.com/few-shot-learning/

[12] https://www.analyticsvidhya.com/blog/2023/07/few-shot-learning/

[13]https://en.wikipedia.org/wiki/Siamese_neural_network

[14]https://towardsdatascience.com/a-friendly-introduction-to-siamese-networks-85ab17522942

[15]https://www.techtarget.com/searchcio/definition/transfer-learning#:~:text=Transfer%20learning%20is%20a%20machine,create%20a%20model%20from%20scratch.

[16]https://datascientest.com/en/transfer-learning-what-is-it

[17]https://link.springer.com/article/10.1007/s10462-021-10004-4

[18] https://arxiv.org/pdf/1703.05175

[19]https://www.researchgate.net/figure/The-classification-process-of-prototypical-network-Conv-denotes-the-convolutional-block_fig2_356072739

[20]https://dcase.community/challenge2024/task-few-shot-bioacoustic-event-detection

[21]https://dcase.community/documents/challenge2022/technical_reports/DCASE2022_Du_122_5.pdf

[22]https://dcase.community/documents/workshop2023/proceedings/DCASE2023Workshop_Moummad_63.pdf

[23]https://www.youtube.com/watch?v=HH_h52I_Qeg

https://www.youtube.com/watch?v=bm1cQfb_pLA

[24] https://guides.slv.vic.gov.au/digitalaudiofundamentals/audioprocessing

All the code created for this work will be found at the repository:

https://github.com/AldoBarca/Thesis