ΔΗΜΟΚΡΙΤΟΣ
ΕΘΝΙΚΟ ΚΕΝΤΡΟ ΕΡΕΥΝΑΣ ΦΥΣΙΚΩΝ ΕΠΙΣΤΗΜΩΝ

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ
UNIVERSITY OF PIRAEUS

Inter-Institutional MSc "Artificial Intelligence"

Assignment on course:

"Deep Learning"

Title:

"Harmonies of Heritage"

FOUKANELIS CHRISTOS (mtn2324)

VYTINIOTIS KONSTANTINOS (mtn2308)

24 June 2024

This page is intentionally left blank

Table of Contents

This page is intentionally left blank

1. Introduction

1.1. Project Overview

In recent years, the rapid advancements in digital technology and machine learning have opened new avenues for the preservation and analysis of cultural heritage. This project aims to leverage deep learning practices, such as Convolutional Neural Networks (CNNs), Recurrent Neural Networks (RNNs), Long Short-Term Memory networks (LSTMs), and Transformers, to classify Greek folk songs by their region of origin. By doing so, we can facilitate the cataloging and analysis of these traditional songs, thereby aiding cultural researchers and enthusiasts in preserving and understanding the rich musical heritage of Greece.

The primary objective of this project is to develop a robust classification system that can accurately identify the regional origins of Greek traditional and folk songs. This system will utilize modern digital tools and techniques to process and analyze audio data, extracting relevant features and patterns that are indicative of specific regional styles.

1.2. Motivation and Goals

Our cultural heritage is deeply rooted in our traditions, including traditional songs from across the diverse regions of Greece and those of Greek influence (such as Asia Minor). These songs embody rich cultural backgrounds and reflect the unique local traditions of their respective regions.

The methods described in this report, aim to streamline the cataloging and analysis of traditional songs, aiding cultural researchers such as the renowned Dora Stratou and Domna Samiou.

Given the extensive repertoire of songs from diverse regions across Greece, it is crucial to utilize modern digital media and tools to preserve and uphold our cultural heritage, ensuring the continuity of cultural records for present and future generations.

1.3. Greek Traditional & Folk music

Greek traditional and folk music is a rich and diverse tapestry that reflects the history, geography, and cultural influences of the regions from which it originates. Each region of Greece

has its own distinct musical style, instruments, rhythms, and lyrical themes, which have been passed down through generations. From the mainland to the islands, and from the mountains to the coastal areas, Greek folk music captures the essence of local traditions and stories.

Key characteristics of Greek traditional and folk music include:

- Instrumentation: Traditional instruments such as the bouzouki, lyra, santouri, and baglamas are commonly used, each contributing unique sounds to the music.
- Rhythms and Melodies: The rhythms and melodies often reflect the dance styles and folk traditions of the region, with variations in tempo, meter, and musical scales.
- Lyrical Themes: The lyrics of folk songs often tell stories of daily life, historical events, love, and nature, providing a window into the cultural and social life of the community.
- Regional Variations: Each region has developed its own musical idioms, influenced by historical events, migrations, and interactions with neighboring cultures.

This diversity makes Greek traditional and folk music a fascinating subject for study and preservation, and underscores the importance of using advanced methods to classify and analyze these cultural artifacts accurately.

2. Data Description

2.1. Dataset Overview

Our dataset is derived from the Lyra dataset, a collection of traditional and folklore Greek songs sourced directly from YouTube videos:

- https://github.com/pxaris/lyra-dataset [1]

It includes YouTube video IDs and the specific timeframes where each song appears within the videos. To begin our project, we automate the process of downloading these videos and extracting the necessary segments to gather raw data for our analysis.

Approximately one-third of the videos in the dataset are currently inaccessible. Therefore, we supplemented our dataset through manual efforts to ensure comprehensive coverage and data integrity.

Additionally, a small portion of the dataset was not labelled with an appropriate region. After going through it (and listening to the audios), we labelled most of these as *Mainland Greece*.

2.2. Data Preprocessing

Data preprocessing is a crucial step in any machine learning project as it ensures that the data is clean, consistent, and suitable for analysis. The preprocessing steps undertaken for this project are as follows:

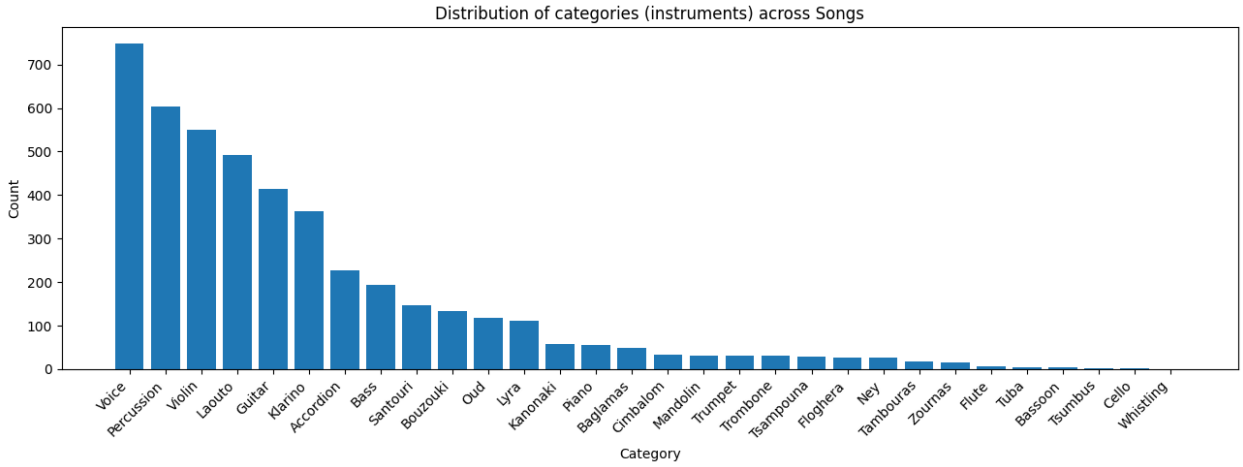
- Removing Invalid Entries: Lyra dataset initially included entries with invalid YouTube IDs, which were not useful for our analysis. We filtered out these invalid entries to ensure that all the remaining entries had valid and accessible audio sources.
- Merging Datasets: After cleaning the Lyra dataset, we merged it with the enhanced dataset, which contained additional audio files that we collected independently. This merging process involved aligning the data formats and ensuring that there were no duplicate entries.
- Creating the Final Dataset: The cleaned and merged data was compiled into a single file, ***gr_folk_music.csv***. This final dataset contains all the relevant information needed for our analysis, including metadata such as the region of origin, song title, and audio source.

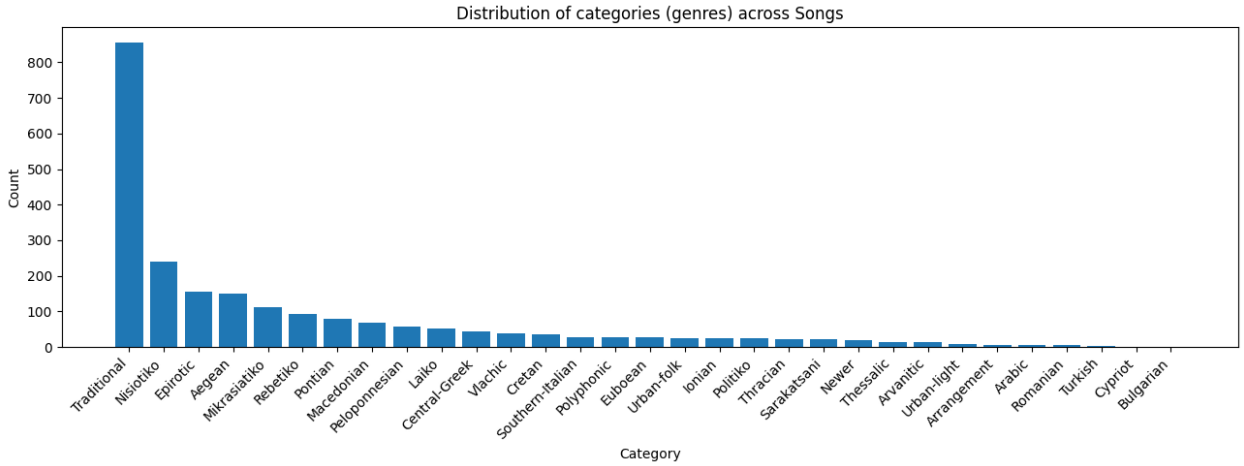
- Grouping Regions: To facilitate more robust analysis and classification, we grouped certain regions together, resulting in 21 large categories. This grouping was based on geographical proximity and cultural similarities, ensuring that each category represented a distinct musical tradition.

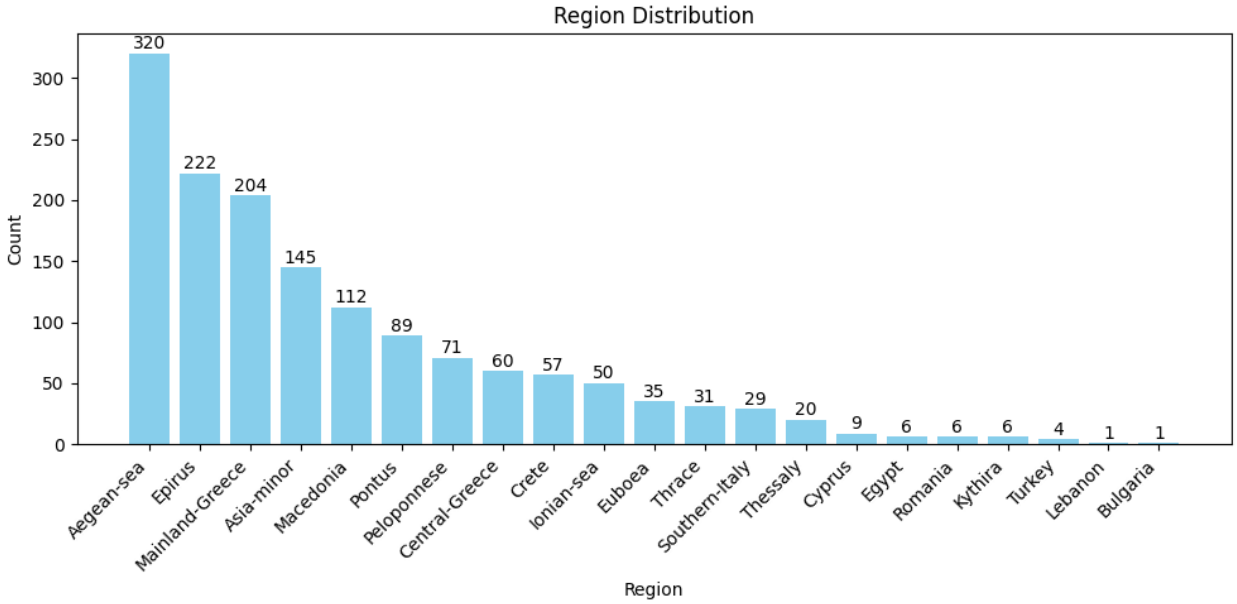
By performing these preprocessing steps, we ensured that the dataset was clean, comprehensive, and ready for the next stages of feature extraction and model training. The final dataset provides a solid foundation for our project, enabling us to explore and analyze the rich diversity of Greek traditional and folk music.

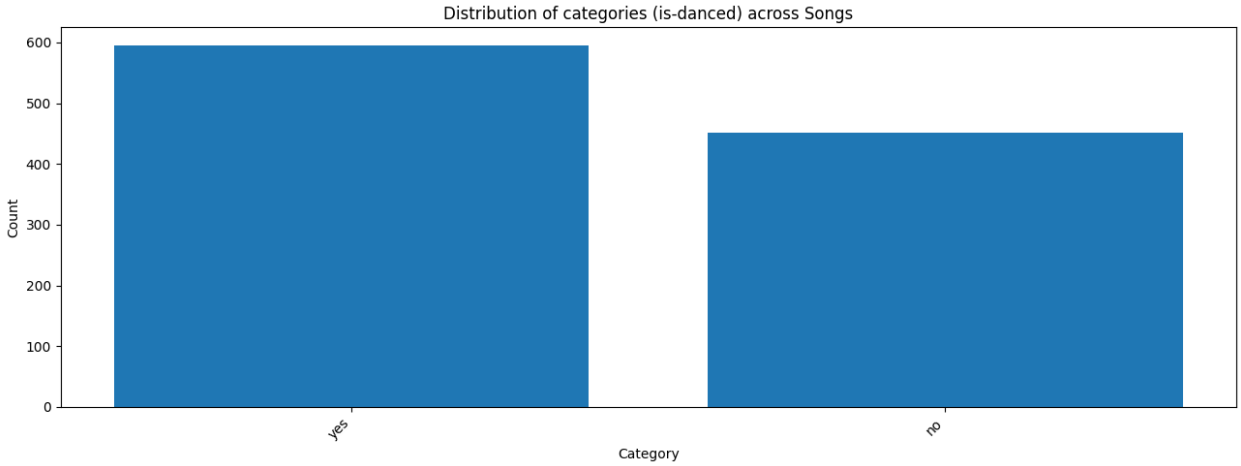
It is imperative to note that the videos from where the audio is coming from contains several sources of noise, such as clapping, whistling, speech and object moving which is bound to affect our results.

2.3. Data Exploration




Distribution of categories (genres) across Songs
Count
Category
Traditional
Nisiotiko
Epirotic
Aegean
Mikrasiatiko
Rebetiko
Pontian
Macedonian
Peloponnesian
Laiko
Central-Greek
Vlachic
Cretan
Southern-Italian
Polyphonic
Euboean
Urban-folk
Ionian
Politiko
Thracian
Sarakatsani
Newer
Thessalic
Arvanitic
Urban-light
Arrangement
Arabic
Romanian
Turkish
Cypriot
Bulgarian


Region Distribution
Count
Region
320
222
204
145
112
89
71
60
57
50
35
31
29
20
9
6
6
6
4
1
1
Aegean-sea
Epirus
Mainland-Greece
Asia-minor
Macedonia
Pontus
Peloponnese
Central-Greece
Crete
Ionian-sea
Euboea
Thrace
Southern-Italy
Thessaly
Cyprus
Egypt
Romania
Kythira
Turkey
Lebanon
Bulgaria


Distribution of categories (is-danced) across Songs
Count
Category
yes
no

3. Feature Extraction

Below we provide a short description of the features we utilized throughout the project.

- Zero Crossing Rate (ZCR): ZCR is the rate at which the audio signal changes sign from positive to negative or vice versa. It provides an indication of the signal's noisiness or the presence of high-frequency content.
- Energy: Energy represents the total magnitude of the audio signal. It is calculated as the sum of squared amplitudes of the signal over a window, reflecting the signal's loudness.
- Energy Entropy: Energy Entropy measures the variation of energy distribution within the signal. High entropy indicates a more uniform energy distribution, often associated with complex or unpredictable signals.
- Spectral Centroid: Spectral Centroid is the "center of mass" of the spectrum and is calculated as the weighted mean of the frequencies present in the signal. It indicates where the "center of gravity" of the spectrum is located and is correlated with the perceived brightness of the sound.
- Spectral Spread: Spectral Spread measures the dispersion of the spectrum around its centroid. It provides information about the bandwidth of the signal and how spread out the frequencies are.
- Spectral Entropy: Spectral Entropy quantifies the complexity or randomness of the spectral distribution. Similar to Energy Entropy, it reflects the unpredictability of the frequency components within the signal.
- Spectral Flux: Spectral Flux measures the rate of change in the power spectrum of the signal. It is used to detect changes in the signal's frequency content over time, indicating the presence of new or vanishing spectral components.
- Spectral Rolloff: Spectral Rolloff is the frequency below which a specified percentage (usually 85%) of the total spectral energy is contained. It helps in distinguishing between harmonic content (like musical tones) and noise.
- Mel-Frequency Cepstral Coefficients (MFCCs): MFCCs are coefficients that collectively represent the short-term power spectrum of a sound. They are widely used in speech and audio processing to capture the timbral texture of the audio signal.
- Chroma Features: Chroma features represent the twelve different pitch classes in the Western music scale, capturing harmonic and melodic characteristics. They are useful for tasks involving harmony and chord recognition.

- Beats Per Minute (BPM): BPM is a measure of the tempo of a piece of music, indicating the number of beats in one minute. It is crucial for rhythm analysis and tempo-related tasks in music processing.
- Ratio: The Ratio feature often refers to the harmonic-to-noise ratio, which measures the proportion of harmonic (periodic) components to noise (aperiodic) components in the signal. It is useful for distinguishing between musical sounds and noise.

All these features collectively provide a comprehensive representation of the audio signal, capturing various aspects of its temporal, spectral, and perceptual characteristics, which are essential for tasks like classification, segmentation, and analysis.

3.1. Feature Extraction with `PyAudioAnalysis`

PyAudioAnalysis is an open-source Python library that provides easy-to-use functionalities for audio signal analysis and feature extraction. It is widely used in the fields of audio processing and machine learning due to its comprehensive set of tools for extracting various audio features and performing audio classification, segmentation, and visualization.

In this project, PyAudioAnalysis was utilized to extract a rich set of features from audio recordings of Greek traditional and folk songs. The extracted features are essential for building machine learning models like Support Vector Machines (SVM) and Feedforward Neural Networks (FFNN) for classifying these songs by their region of origin.

The process of feature extraction using *PyAudioAnalysis* involves several key steps:

- Process audio files stored in different directories, each representing a specific region.
- Compute a variety of audio features for each audio file and aggregate them to provide a comprehensive feature set.
- For each audio file, the mean and standard deviation of each feature were computed, resulting in a comprehensive feature vector that summarizes the audio signal's characteristics. These aggregated features were combined to create a final dataset with 138 features per song, including both mean and standard deviation values for each primary feature.
- Labeling: Each feature vector was labeled with the corresponding region of the song, enabling supervised learning algorithms to use these labeled data points for training and classification tasks.

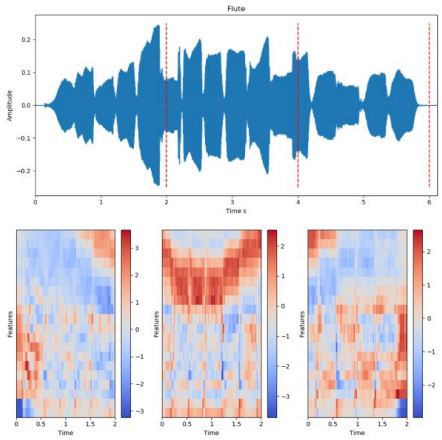
The extracted features were saved in a CSV file (***ml_features.csv***) and a serialized file (***ml_features.pkl***) for easy access and reuse. These features were then used in our baseline as will be explained in the next chapters.to train and evaluate machine learning models:

3.2. Feature Extraction with `librosa`

In audio classification, particularly in the context of music, it is a common practice to utilize meticulously hand-crafted features [7]. These features are specifically designed to capture the unique and intricate characteristics of the audio in a robust manner. By focusing on aspects such as timbre, rhythm, melody, and harmony, these features can significantly enhance the accuracy and effectiveness of the classification process. Commonly used features include Mel-Frequency Cepstral Coefficients (MFCCs), which represent the short-term power spectrum of sound and are effective in capturing the timbral texture, and chroma features, which capture the pitch content and harmonic aspects of the music. By leveraging domain-specific knowledge to create representations that are more informative and discriminative, these hand-crafted features lead to better performance compared to using raw audio data alone.

The feature-based approach shares some similarities with the raw data-based approach on the aspect of segmentation and down sampling. In detail, each song was converted to a non-fixed amount of short-term windows (a couple of ms each) according to its length. All the features (mfcc, chroma etc.) were calculated for each short-term window and after that these windows were grouped dynamically to form a constant length of 100 lists of windows (segments). These segments contained several short-term windows that were averaged to produce a single value for each respective feature. No further padding is required because the down-sampling handled that (when the windows where grouped and truncated). So, the model on each iteration will have to process an input of shape (batch size x 100 x 28) where 100 are the number of audio segments and 28 are the total averaged features for each segment.

Below is an example visualization from another audio classification task that better illustrates the segmentation and averaging process [3][6].


Flute
Amplitude
Time s
Features
Time

4. Baseline

4.1. SVM

The SVM model utilizes the extracted features to learn the boundaries between different regional categories. SVM is particularly effective for high-dimensional feature spaces and can handle the complex relationships between the extracted audio features.

Using Recursive Feature Elimination (RFE), the hyperparameters chosen were:

- C: 1000
- gamma: 0.01
- kernel: rbf

Splits:

- Train-Validation / Test split: 80% – 20%
- Train / Validation split: 80% – 20%

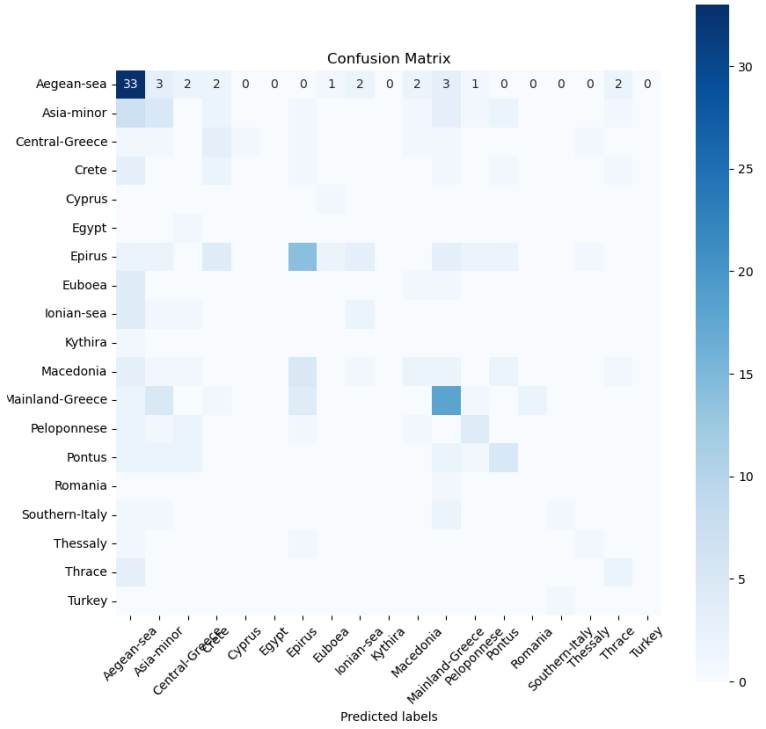
Trained model stats:

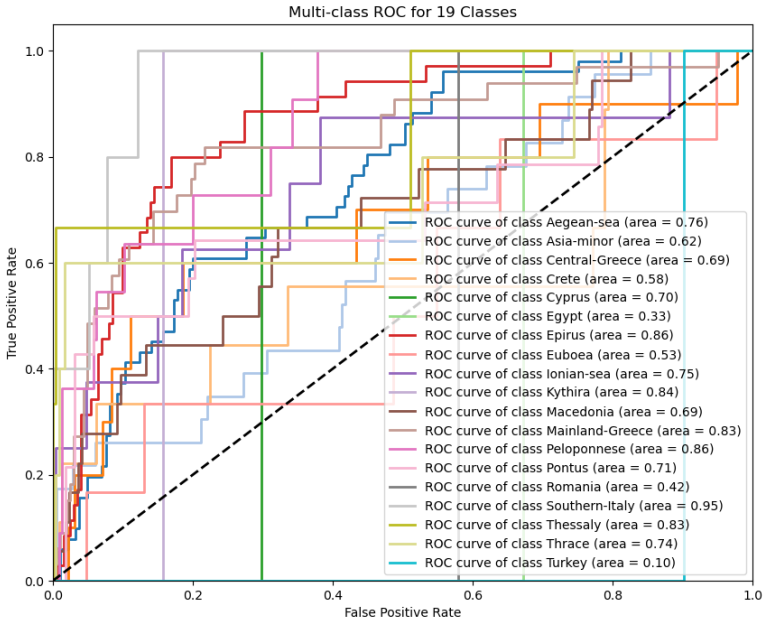
- Accuracy: 0.377
- Precision: 0.225: This indicates that there are many false positives, suggesting that the model is not very precise in its predictions for individual classes.
- Recall: 0.213: A recall of 0.213 means that the model correctly identified 21.3% of all instances that belong to each class. This indicates that the model has a high number of false negatives and is missing a significant portion of the actual instances for each class.
- F1: 0.212: This indicates that the balance between precision and recall is quite low, indicating poor overall performance in both detecting and accurately predicting the correct classes.

Dataset used:

- Features extracted from *PyAudioAnalysis* (138 features)

Validation Results





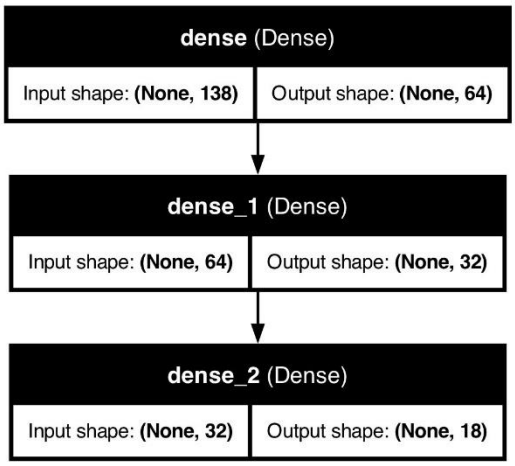
Results using the test dataset

Classification Report:	precision	recall	f1-score	support
Aegean-sea	0.45	0.53	0.49	64
Asia-minor	0.42	0.45	0.43	29
Central-Greece	0.29	0.50	0.36	12
Crete	0.15	0.18	0.17	11
Cyprus	0.00	0.00	0.00	2
Egypt	0.00	0.00	0.00	1
Epirus	0.44	0.53	0.48	45
Euboea	0.00	0.00	0.00	7
Ionian-sea	0.00	0.00	0.00	10
Kythira	0.00	0.00	0.00	1
Macedonia	0.54	0.30	0.39	23
Mainland-Greece	0.59	0.49	0.53	41
Peloponnese	0.40	0.29	0.33	14
Pontus	0.67	0.56	0.61	18
Romania	0.00	0.00	0.00	1
Southern-Italy	0.50	0.50	0.50	6
Thessaly	0.00	0.00	0.00	4
Thrace	0.25	0.17	0.20	6
Turkey	0.00	0.00	0.00	1
accuracy			0.42	296
macro avg	0.25	0.24	0.24	296
weighted avg	0.42	0.42	0.41	296

4.2. Feedforward NN

Feedforward Neural Network (FFNN):

The FFNN model uses the feature vectors as input to learn patterns and relationships in the data through multiple layers of neurons. This model is flexible and can capture non-linear dependencies in the data, making it suitable for the task of music classification.



Training stats:

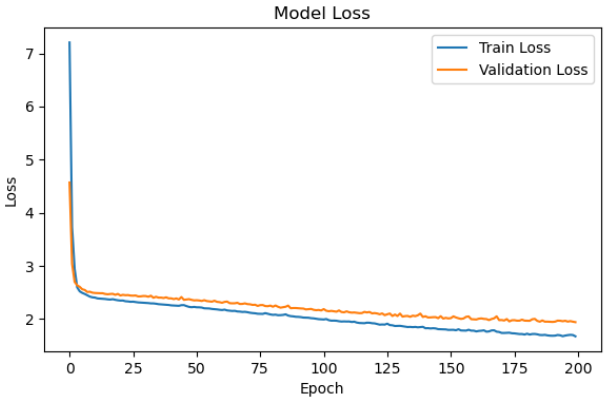
- EPOCHS = 200
- BATCH_SIZE = 100

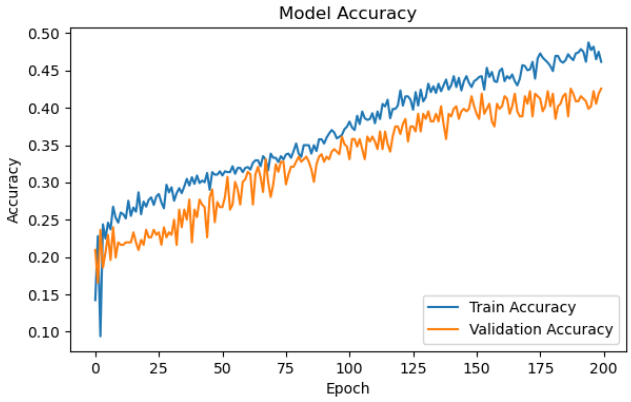
Dataset used:

- Features extracted from *PyAudioAnalysis* (138 features)

Training results:

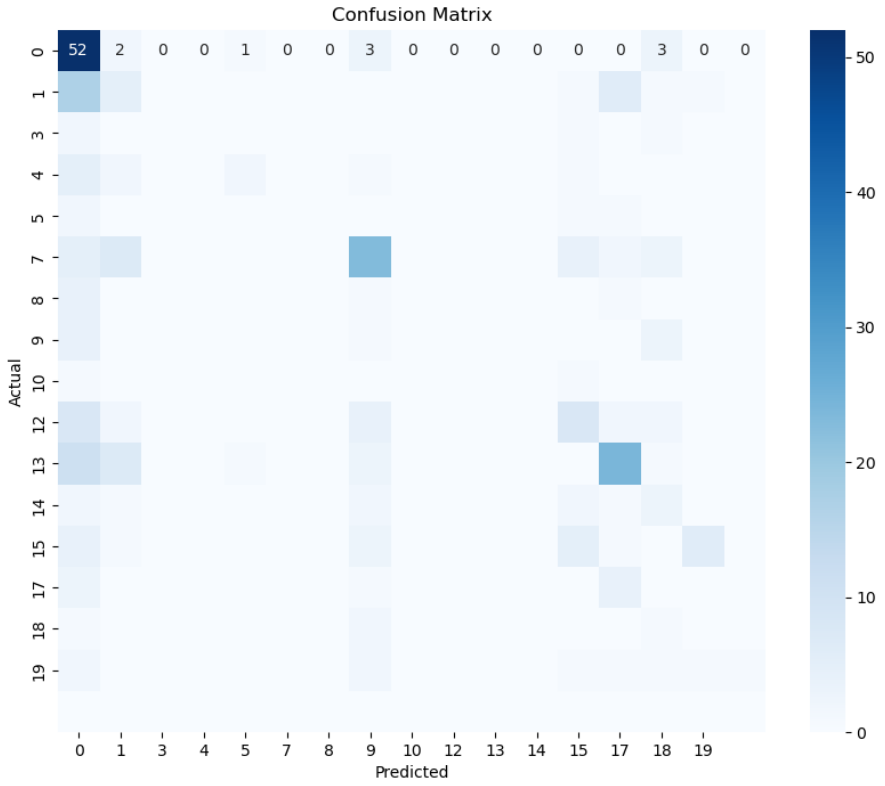
- Loss: 1.94, Accuracy: 0.425





Results using the test dataset

- Test Accuracy: 0.234



Results Interpretation

Training/Validation Loss Graph

- Convergence: Both the training and validation loss decrease steadily over epochs, indicating that the model is learning and converging. This is a positive sign showing that the model is optimizing and reducing error over time.
- Overfitting: The training loss is slightly lower than the validation loss, especially in the early stages. However, towards the later epochs, the difference between the training and validation loss remains relatively small, suggesting that overfitting is not severe.

The model seems to generalize reasonably well to the validation data.

Training/Validation Accuracy Graph

- Both training and validation accuracy increase over time, indicating that the model is learning and improving its predictive capabilities.
- The training accuracy is consistently higher than the validation accuracy.
- The accuracy curves do not show significant divergence, which suggests the model has not yet hit a severe overfitting regime.
- The validation accuracy fluctuates but follows the trend of the training accuracy, indicating the model's predictions on the validation set are fairly consistent with those on the training set.

Confusion Matrix of test

- Class Imbalance: The confusion matrix shows a strong diagonal for a few classes, indicating good performance for those classes. Some classes have very few correctly classified instances, which might indicate a class imbalance or difficulty in distinguishing these classes.
- Misclassifications: The matrix shows several off-diagonal elements, indicating misclassifications where instances of one class are predicted as another.
- Specific classes (e.g., class 1) have a high number of correct predictions (52) compared to others, suggesting that the model performs well for these classes but struggles with others.
- Confusion Between Specific Classes: There are some noticeable confusions (e.g., class 9 being predicted as class 13 and vice versa), suggesting that the features distinguishing these classes are not being effectively learned by the model. This might be due to similarities between these classes or insufficient distinctive features being captured during training.

Improvements and future steps

- Address class imbalance by augmenting the data or using techniques like class weighting.
- Enhance feature extraction or use more advanced architectures to capture better distinguishing features.
- Further tune hyperparameters or consider using ensemble methods to boost performance.

5. Spectrograms and CNNs

5.1. Creating spectrograms and mel-spectrograms

Spectrograms and mel-spectrograms are essential tools in audio analysis and machine learning. They provide a visual representation of the frequency spectrum of a signal as it varies with time, which is particularly useful for training Convolutional Neural Networks (CNNs) for audio classification tasks. This section details the process used to generate these spectrograms and mel-spectrograms from our audio dataset.

5.1.1. Spectrograms

To get into more details, a spectrogram is a visual representation of the spectrum of frequencies in a sound signal as they vary with time. The following steps were undertaken to create spectrograms from the audio files:

- Libraries: Necessary libraries such as librosa, matplotlib, and *numpy* were used.
- Class Initialization: Defined a Spectrogram class that takes a list of audio file paths. Ensure the output directory for spectrograms exists.
- Audio Signal Normalization: Normalized the audio signal to ensure consistent amplitude scaling. Converted the signal to double precision and normalized it based on its mean and maximum absolute value.
- Spectrogram Generation: Compute the Short-Time Fourier Transform (STFT) of the audio signal. Converted the amplitude to decibels (dB) to obtain the spectrogram.
- Spectrogram Plotting: Used Librosa to plot the spectrogram. Saved the plot as a PNG file without axis labels and color bars to focus on the spectrogram image.
- Memory Management: Due to memory leaks from the different libraries and Python's way of handling memory, the process could not be initially automated as it would crash after just 10 plot generations. Performed smart garbage collection to free up memory after processing each file.

Fig. 1, Spectrogram – Aegean Sea

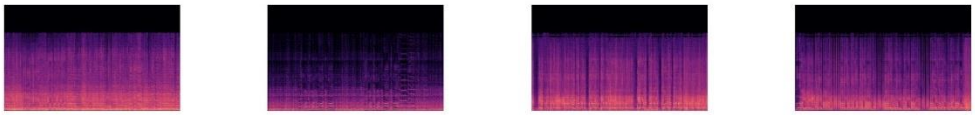


Fig. 2, Spectrogram – Asia Minor

Fig. 3, Spectrogram – Central Greece

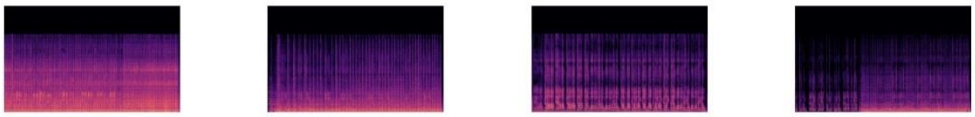


Fig. 4, Spectrogram – Crete

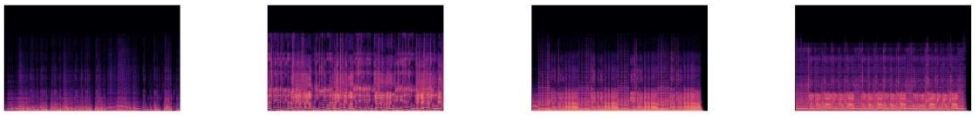


Fig. 5, Spectrogram – Cyprus

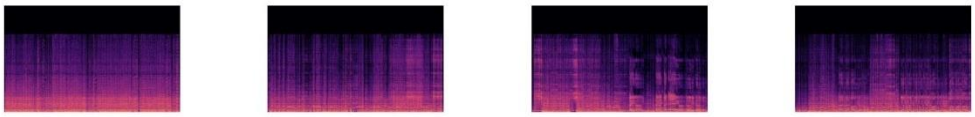


Fig. 6, Spectrogram – Egypt

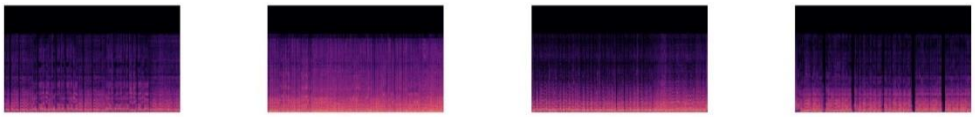


Fig. 7, Spectrogram – Epirus

Fig. 8, Spectrogram – Euboea

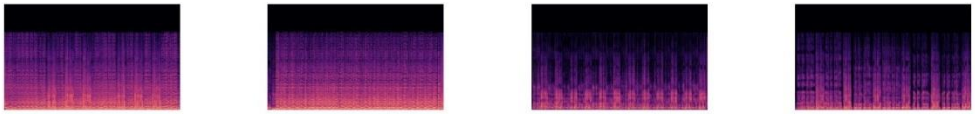


Fig. 9, Spectrogram – Ionian Sea

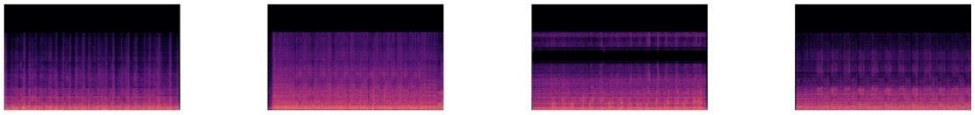


Fig. 10, Spectrogram – Kythira

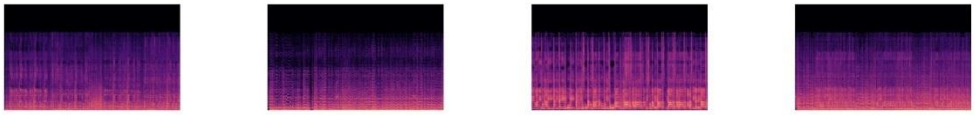


Fig. 11, Spectrogram – Macedonia

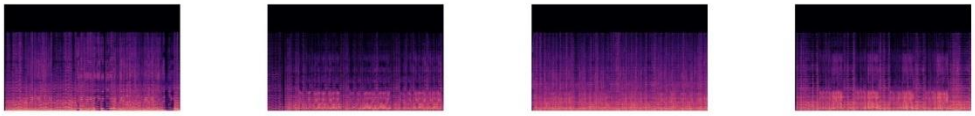


Fig. 12, Spectrogram – Mainland Greece

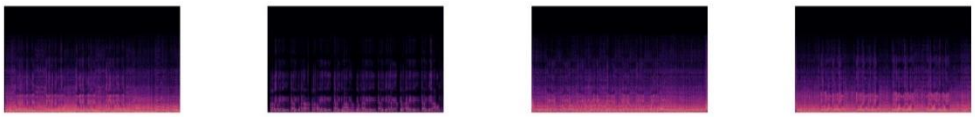


Fig. 13, Spectrogram – Peloponnese

Fig. 14, Spectrogram – Pontus

Fig. 15, Spectrogram – Romania

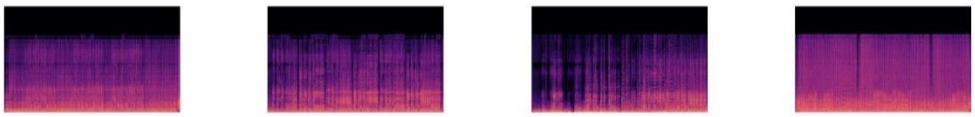


Fig. 16, Spectrogram – Southern Italy

Fig. 17, Spectrogram – Thessaly

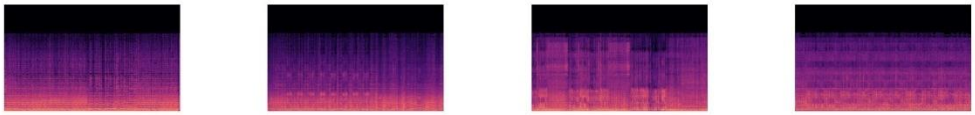


Fig. 18, Spectrogram – Thrace

5.1.2. Mel-Spectrograms

A mel-spectrogram is a type of spectrogram where the frequencies are converted to the mel scale, which more closely approximates the human ear's response to different frequencies. The process to create mel-spectrograms is similar to creating regular spectrograms with some modifications:

- Computed Mel-Spectrogram: After loading and normalizing the audio signal, compute the mel-spectrogram using librosa.feature.melspectrogram.
- Converted the mel-spectrogram to dB using librosa.power_to_db.
- Similar to the spectrogram extraction, plotted and saved mel-spectrograms as PNG files without axis labels and color bars to focus on the spectrogram image.

Fig. 19, Mel Spectrogram – Aegean Sea

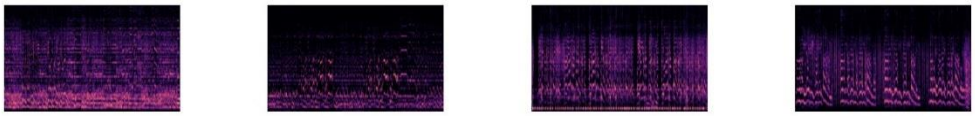


Fig. 20, Mel Spectrogram – Asia Minor

Fig. 21, Mel Spectrogram – Central Greece

Fig. 22, Mel Spectrogram – Crete

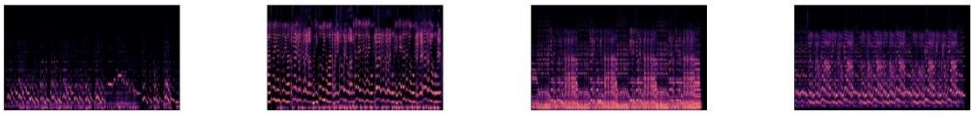


Fig. 23, Mel Spectrogram – Cyprus

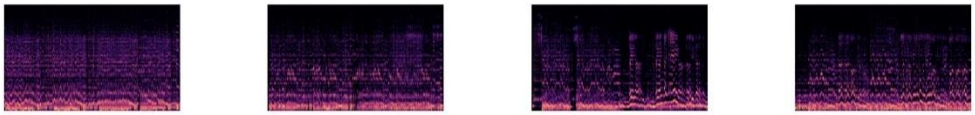


Fig. 24, Mel Spectrogram – Egypt

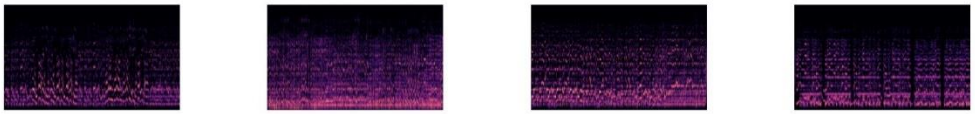


Fig. 25, Mel Spectrogram – Epirus

Fig. 26, Mel Spectrogram – Euboea

Fig. 27, Mel Spectrogram – Ionian Sea

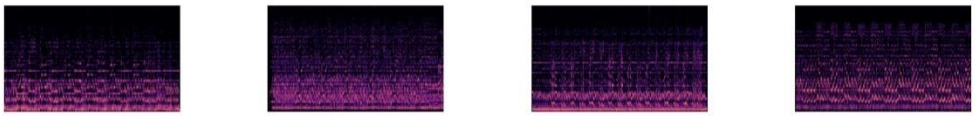


Fig. 28, Mel Spectrogram – Kythira

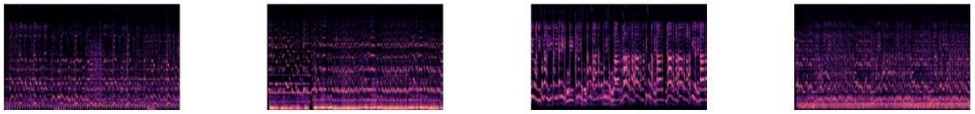


Fig. 29, Mel Spectrogram – Macedonia

Fig. 30, Mel Spectrogram – Mainland Greece

Fig. 31, Mel Spectrogram – Peloponnese

Fig. 32, Mel Spectrogram – Pontus

Fig. 33, Mel Spectrogram – Romania

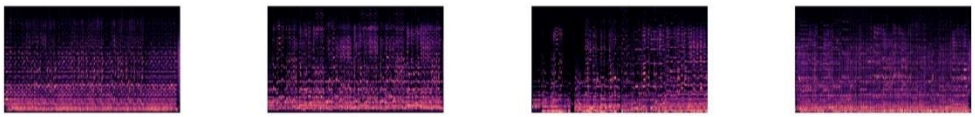


Fig. 34, Mel Spectrogram – Southern Italy

Fig. 35, Mel Spectrogram – Thessaly

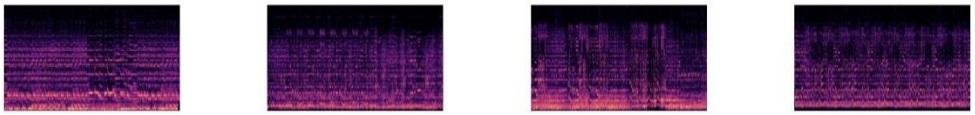


Fig. 36, Mel Spectrogram – Thrace

Upon examining the (mel-)spectrograms, it is apparent that most classes are not easily distinguishable from each other and contain quite a bit of noise. This is understandable since the audios are coming from YouTube videos which are not song albums, but, TV programs. Clapping, whistling, speech and object moving are the main sources of noise.

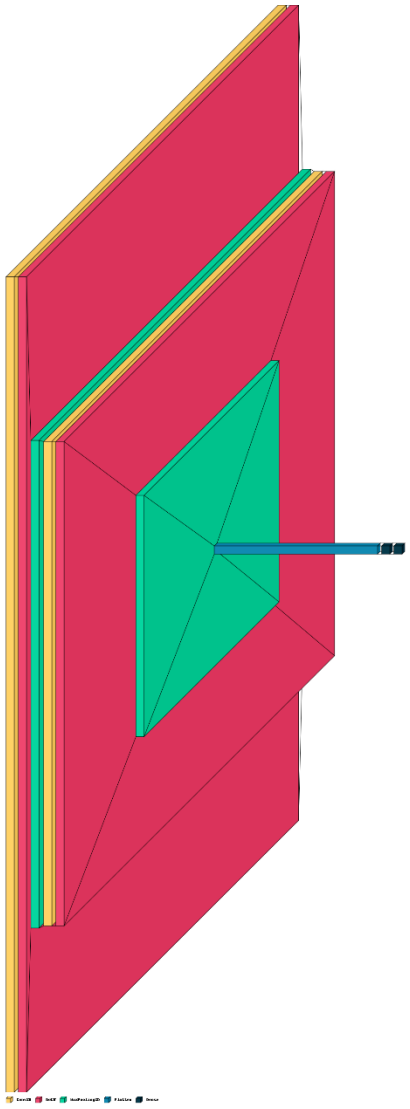
5.2. Simple CNN

Before feeding the images (spectrograms and mel-spectrograms) into the CNN, a series of transformations are applied to augment the data and normalize it. These transformations help in enhancing the model's robustness and performance:

- *ColorJitter*: Adjusts the brightness, contrast, saturation, and hue of the images with a factor of 0.2, introducing variability and helping the model generalize better.
- *ToTensor*: Converts the image to a PyTorch tensor, which is a necessary step for PyTorch models.
- *Normalization*: Normalizes the tensor using the provided mean and standard deviation (mean and std), which helps in stabilizing and speeding up the training process. The mean and std are calculated based on the whole dataset.

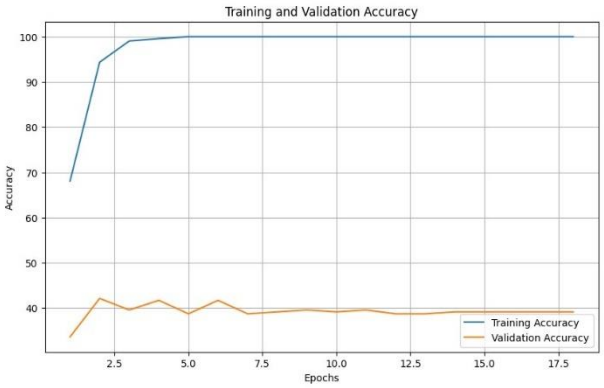
The same image transformations were applied to all PNG before being used by either of the CNNs.

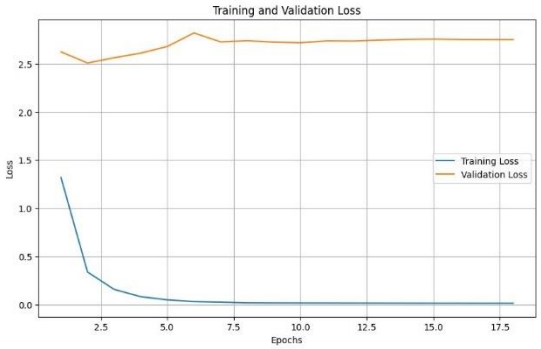
The `SimpleCNN` model is a straightforward Convolutional Neural Network (CNN) designed to classify spectrograms and mel-spectrograms into multiple classes. Here are the details of the architecture:



This SimpleCNN architecture provides a foundational understanding of how convolutional layers and fully connected layers can be combined to classify audio-based images. The model can be further refined and expanded to improve accuracy and generalization capabilities.

Input: Spectograms





Interpretation:

- Training Accuracy: The training accuracy increases rapidly and reaches 100% very quickly. This indicates that the model is able to perfectly classify the training data.
- Validation Accuracy: The validation accuracy fluctuates around 40% and does not show significant improvement after the initial few epochs.

Key Observations:

- Overfitting: The model exhibits a classic case of overfitting. It performs exceptionally well on the training data but fails to generalize to the validation data, as indicated by the substantial gap between training and validation accuracy.
- Early Convergence: The training accuracy reaches its maximum almost immediately, suggesting that the model has memorized the training data rather than learning generalizable patterns.

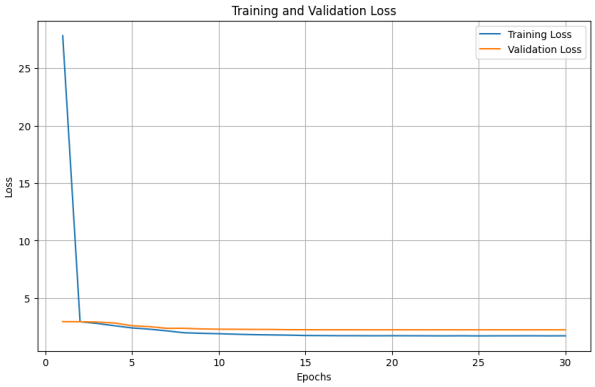
Results on test dataset reflect the above observations

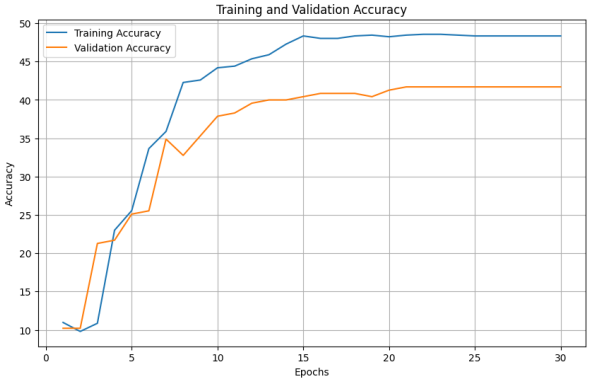
```
Accuracy: 0.1689
Precision: 0.1183
Recall: 0.1689
F1 Score: 0.0872
                precision    recall  f1-score   support

     Aegean-sea       0.38      0.14      0.20        64
     Asia-minor       0.00      0.00      0.00        29
 Central-Greece       0.00      0.00      0.00        12
          Crete       0.00      0.00      0.00        12
         Cyprus       0.00      0.00      0.00         2
          Egypt       0.00      0.00      0.00         2
         Epirus       0.15      0.91      0.26        45
         Euboea       0.00      0.00      0.00         7
     Ionian-sea       0.00      0.00      0.00        10
        Kythira       0.00      0.00      0.00         2
      Macedonia       0.00      0.00      0.00        23
Mainland-Greece       0.00      0.00      0.00        41
    Peloponnese       0.33      0.07      0.11        15
         Pontus       0.00      0.00      0.00        18
        Romania       0.00      0.00      0.00         2
 Southern-Italy       0.00      0.00      0.00         6
       Thessaly       0.00      0.00      0.00         4
         Thrace       0.00      0.00      0.00         7
         Turkey       0.00      0.00      0.00         1

       accuracy                           0.17       302
      macro avg       0.05      0.06      0.03       302
   weighted avg       0.12      0.17      0.09       302
```

Improvements and future steps

- Regularization: Implement techniques such as dropout, L2 regularization, or data augmentation to prevent overfitting.
- Early Stopping: Use early stopping based on validation performance to prevent the model from overfitting.

Input: Mel-Spectograms





Interpretation

- Training Accuracy: The training accuracy shows a steady increase over the epochs, eventually reaching around 50%. This indicates that the model is learning from the training data and improving its performance over time.
- Validation Accuracy: The validation accuracy also improves, but it stabilizes around 40%. While there is improvement, it is not as high as the training accuracy.
- Training Loss: The training loss decreases rapidly in the initial epochs and then stabilizes at a relatively low value. This indicates that the model is quickly learning the training data and minimizing the loss.

Key Observations

- Learning Trend: The training accuracy increases consistently, indicating that the model is effectively learning the training data.
- Stabilization: Both training and validation accuracy curves start to stabilize towards the end of the epochs, indicating that the model is converging and not significantly improving further.

Results on test dataset reflect the above observations

```
Accuracy: 0.3344
Precision: 0.2357
Recall: 0.3344
F1 Score: 0.2366
                precision    recall  f1-score   support

     Aegean-sea       0.33      0.95      0.49        64
     Asia-minor       0.00      0.00      0.00        29
 Central-Greece       0.00      0.00      0.00        12
          Crete       0.00      0.00      0.00        12
         Cyprus       0.00      0.00      0.00         2
          Egypt       0.00      0.00      0.00         2
         Epirus       0.63      0.38      0.47        45
         Euboea       0.00      0.00      0.00         7
     Ionian-sea       0.00      0.00      0.00        10
        Kythira       0.00      0.00      0.00         2
      Macedonia       0.00      0.00      0.00        23
Mainland-Greece       0.27      0.49      0.34        41
    Peloponnese       0.00      0.00      0.00        15
         Pontus       0.60      0.17      0.26        18
        Romania       0.00      0.00      0.00         2
 Southern-Italy       0.00      0.00      0.00         6
       Thessaly       0.00      0.00      0.00         4
         Thrace       0.00      0.00      0.00         7
         Turkey       0.00      0.00      0.00         1

       accuracy                           0.33       302
      macro avg       0.10      0.10      0.08       302
   weighted avg       0.24      0.33      0.24       302
```

Conclusions

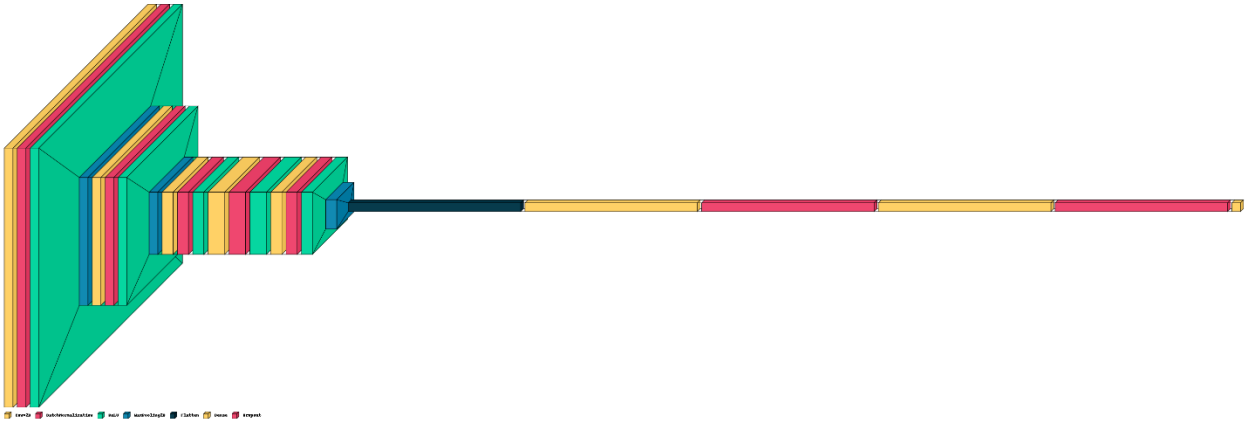
The consistent gap between training and validation accuracy, along with the closeness of the loss curves, indicates that while the model learns well, there is still room to improve its generalization to unseen data. Incorporating regularization methods (such as dropout, L2 regularization) and using data augmentation can help address the slight overfitting observed in the accuracy plot. Incorporating regularization methods (such as dropout, L2 regularization) and using data augmentation can help address the slight overfitting observed in the accuracy plot.

Finally, from the results of this CNN we can conclude that feeding the CNN with mel-spectrograms outperforms the model with regular spectrograms in terms of both training and validation accuracy. That is because mel-spectrograms provide richer and more informative features, leading to better generalization as evidenced by higher validation accuracy and closely aligned loss curves.

5.3. Advanced `CNN` Architectures

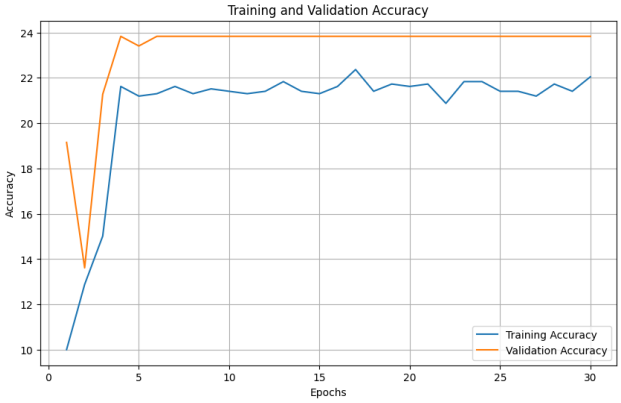
AlexNet is one of the most well-known CNN architectures, developed by Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton. It significantly contributed to the advancement of deep learning by winning the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) in 2012. AlexNet demonstrated that deep learning could outperform traditional computer vision techniques by a large margin.

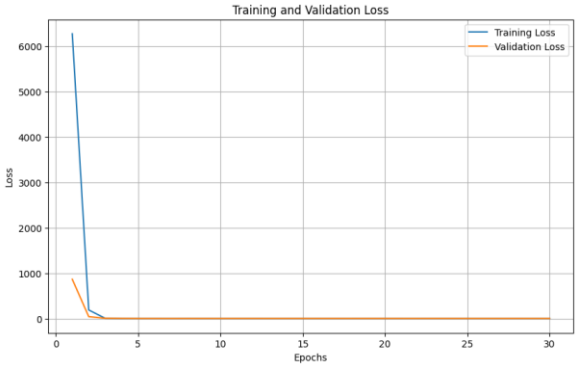
We have used a modified version of the AlexNet model which follows a similar structure to the original AlexNet but is adapted for a specific number of classes [2] and incorporates batch normalization layers for improved training stability and performance.



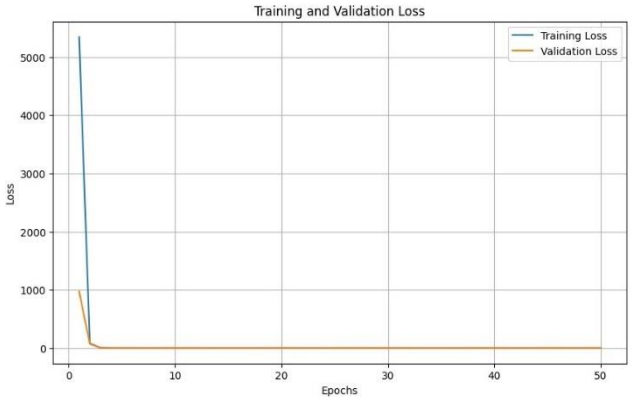
Hyperparameters and input is similar to the aforementioned Simple CNN.

Input: Mel-Spectograms





Input: Spectograms



Results on test dataset (spectrograms)

```
Accuracy: 0.1391
Precision: 0.0300
Recall: 0.1391
F1 Score: 0.0448
                precision    recall  f1-score   support

     Aegean-sea       0.00      0.00      0.00        64
     Asia-minor       0.00      0.00      0.00        29
 Central-Greece       0.00      0.00      0.00        12
          Crete       0.00      0.00      0.00        12
         Cyprus       0.00      0.00      0.00         2
          Egypt       0.00      0.00      0.00         2
         Epirus       0.00      0.00      0.00        45
         Euboea       0.00      0.00      0.00         7
     Ionian-sea       0.00      0.00      0.00        10
        Kythira       0.00      0.00      0.00         2
      Macedonia       0.15      0.17      0.16        23
Mainland-Greece       0.14      0.93      0.24        41
    Peloponnese       0.00      0.00      0.00        15
         Pontus       0.00      0.00      0.00        18
        Romania       0.00      0.00      0.00         2
 Southern-Italy       0.00      0.00      0.00         6
       Thessaly       0.00      0.00      0.00         4
         Thrace       0.00      0.00      0.00         7
         Turkey       0.00      0.00      0.00         1

       accuracy                           0.14       302
      macro avg       0.02      0.06      0.02       302
   weighted avg       0.03      0.14      0.04       302
```

Results on test dataset (mel-spectrograms)

```
Accuracy: 0.2119
Precision: 0.0449
Recall: 0.2119
F1 Score: 0.0741
                precision    recall  f1-score   support

     Aegean-sea       0.21      1.00      0.35        64
     Asia-minor       0.00      0.00      0.00        29
 Central-Greece       0.00      0.00      0.00        12
          Crete       0.00      0.00      0.00        12
         Cyprus       0.00      0.00      0.00         2
          Egypt       0.00      0.00      0.00         2
         Epirus       0.00      0.00      0.00        45
         Euboea       0.00      0.00      0.00         7
     Ionian-sea       0.00      0.00      0.00        10
        Kythira       0.00      0.00      0.00         2
      Macedonia       0.00      0.00      0.00        23
Mainland-Greece       0.00      0.00      0.00        41
    Peloponnese       0.00      0.00      0.00        15
         Pontus       0.00      0.00      0.00        18
        Romania       0.00      0.00      0.00         2
 Southern-Italy       0.00      0.00      0.00         6
       Thessaly       0.00      0.00      0.00         4
         Thrace       0.00      0.00      0.00         7
         Turkey       0.00      0.00      0.00         1

       accuracy                           0.21       302
      macro avg       0.01      0.05      0.02       302
   weighted avg       0.04      0.21      0.07       302
```

Interpretation

- The training and validation accuracy for both spectrograms and mel-spectrograms exhibit a similar trend, with early stabilization around 22% and 24% respectively.
- Both inputs result in moderate learning capability, with no significant overfitting observed.
- The validation loss decreases sharply initially and stabilizes at a low value, closely following the training loss curve. This suggests good generalization.

Key Observations

- Both the accuracy and loss plots for the Modified AlexNet with both spectrogram and mel-spectrogram inputs show early convergence and stabilization. This means the model quickly reaches a point where further training does not significantly improve performance.

Improvements and future steps

- Increasing the dropout rate in fully connected layers may help the model generalize better.
- L2 regularization can penalize large weights, encouraging the model to find simpler solutions that generalize better.
- Increasing the number of filters in convolutional layers can help capture more features.

6. Sequential Architectures (RAW/feature based)

In these approaches we leverage the sequential nature of audio data by passing them through sequential model architectures, namely RNN's which are widely used in deep learning applications regarding audio data [4][5]. Detailed analysis of each architecture will be presented later in the report. The data in all approaches were split in train (70% of total data), validation (15% of total data data) and test sets. The data were split in respect to the class labels in order to maintain a balanced distribution of class appearances in all three sets.

6.1. Approaches

RAW Based

The raw audio samples were used initially for the model training to assess the performance potential of a raw-data based approach. This approach has been previously used in the domain of audio classification but for shorter and less complex audio samples (e.g. heartbeat analysis [8]).

Each audio file (song) was loaded as a '.wav' file and the sample array was extracted using the librosa package. The sample array is an audio time series array containing all audio samples. By default, in the case of stereo audio (left and right audio channel) librosa averages each audio channel and returns a 1D array of the averaged channels as a mono channel. For simplicity, the mono channel representation was used for each song. No scaling / standardization was performed on the raw audio data in order to retain as much of the original information as possible.

After loading each file, the sample array was then segmented to smaller sub-arrays (windows). Each window represents 1 second of song duration and contains n samples, where n is the sampling rate of the audio. For consistency reasons, the sampling rate of 44,1 kHz was used for all the audio files meaning that each audio segment will contain 44100 samples. The frequency of 44,1 kHz. The most widely used sample rate for modern songs. The model will traverse through the windows one by one processing 1 second of audio each time. Additionally, for consistency and robustness, each audio file was wither padded or truncated to match a fixed length of 4,000,000 samples (about 91 seconds which sits at the higher end of audio durations in the dataset). The network processes each file internally un-padded, while the loss and features are also calculated for the un-padded audio files padding was added to facilitate the batching process. So, the model on each iteration will have to process an input of shape (batch size x 91 x 44100) where 91 are the number of audio segments.

This technique is widely used in audio analysis and has proven very effective in aiding model generalization as it provides the model with a constant number of homogenous inputs because each segment has similar audio features in the context of a second, giving the model the ability to extract the underlying features and patterns more easily. Additionally, audio segmenting is computationally and memory efficient because it breaks down the input the network has to process on each iteration. This is crucial for real-world audio processing applications that require low latency and good generalization because in the case of songs, it is very likely that the model will receive an unseen audio sample.

Feature Based

In audio classification, particularly in the context of music, it is a common practice to utilize meticulously hand-crafted features [7]. These features are specifically designed to capture the unique and intricate characteristics of the audio in a robust manner. By focusing on aspects such as timbre, rhythm, melody, and harmony, these features can significantly enhance the accuracy and effectiveness of the classification process. Commonly used features include Mel-Frequency Cepstral Coefficients (MFCCs), which represent the short-term power spectrum of sound and are effective in capturing the timbral texture, and chroma features, which capture the pitch content and harmonic aspects of the music. By leveraging domain-specific knowledge to create representations that are more informative and discriminative, these hand-crafted features lead to better performance compared to using raw audio data alone

6.2. RNN Architectures

Recurrent Neural Networks (RNNs) are commonly used for audio classification due to their ability to effectively handle sequential data, as they are able to capture the temporal and contextual dependencies of audio samples. Additionally, they are able to handle variable length-inputs which is the case exactly for traditional songs.

RNN Architecture Exploration

All three typical RNN models were implemented with various parameter combinations to evaluate their impact on performance. Long Short-Term Memory (LSTM) units and Gated Recurrent Units (GRUs) are inherently designed to retain complex and long-term features, making them the primary focus of these experiments.

Hyperparameters

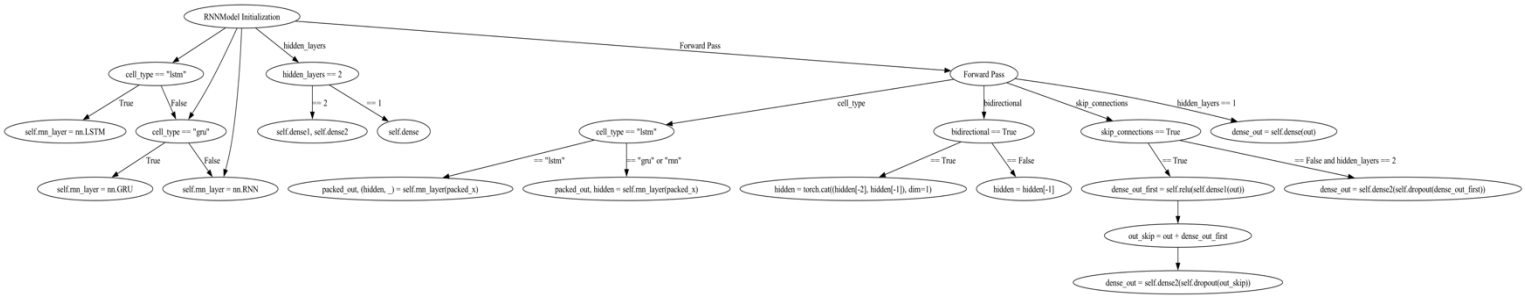
- Learning Rate: Two primary learning rates were tested: 0.001 and 0.0001. Additionally, a learning rate scheduler was applied, reducing the rate by 10% every 10 epochs.
- Number of Stacked RNN Layers: Models were evaluated with 1 and 2 stacked RNN layers to assess their ability to capture hierarchical and complex dependencies, which may require deeper architectures.
- Hidden FF Dense Layers: Two configurations of feed-forward layers (1 and 2) were implemented to study their impact on feature mapping to corresponding classes.
- Skip Connections: For models with 2 feed-forward layers, skip connections were tested. This technique connects the output of the RNN layer directly to the second feed-forward layer.
- Dropout Probability: Various dropout probabilities (0, 0.2, 0.5) were tested to improve generalization and prevent overfitting.
- Gradient Clipping: Gradient clipping, set to True or False, was implemented to mitigate the vanishing gradient problem. It restricts gradients to a range of [-5, 5] during training.
- Cell Type: Different RNN architectures (RNN, LSTM, GRU) were compared to analyze their performance in sequential data tasks.
- Hidden Size of RNN: Experiments were conducted using RNN hidden sizes of 128 and 64 neurons to evaluate their impact on model performance. No more than 128 neurons were used because the model would be in danger of underfitting, due the lack of large-scale data.

- Bidirectional RNN: Bidirectional RNNs were tested to determine if leveraging both forward and backward directions enhances the model's ability to capture intricate features.

The RNN architecture that was used in both the raw data and the feature-based approaches is as follows:

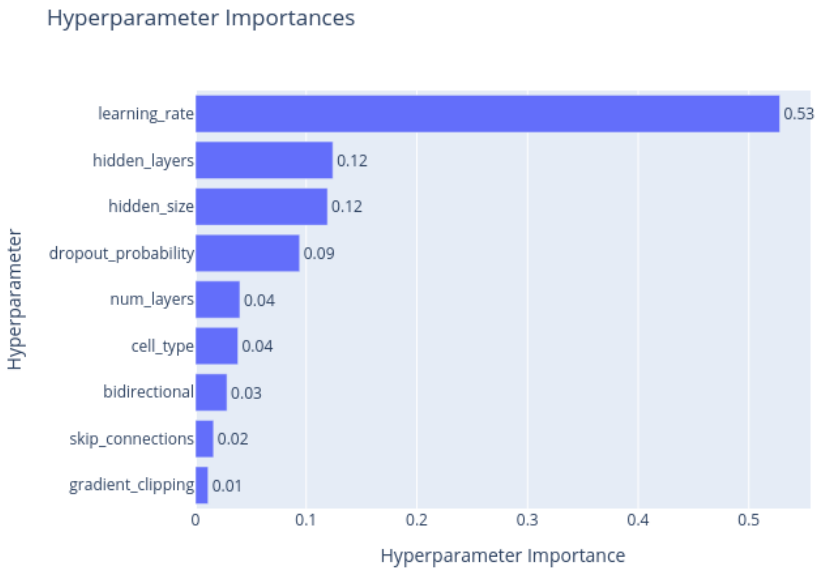
Additionally, early stopping was implemented during training to prevent overfitting. This technique ended the training loop if the training and validation losses converged.

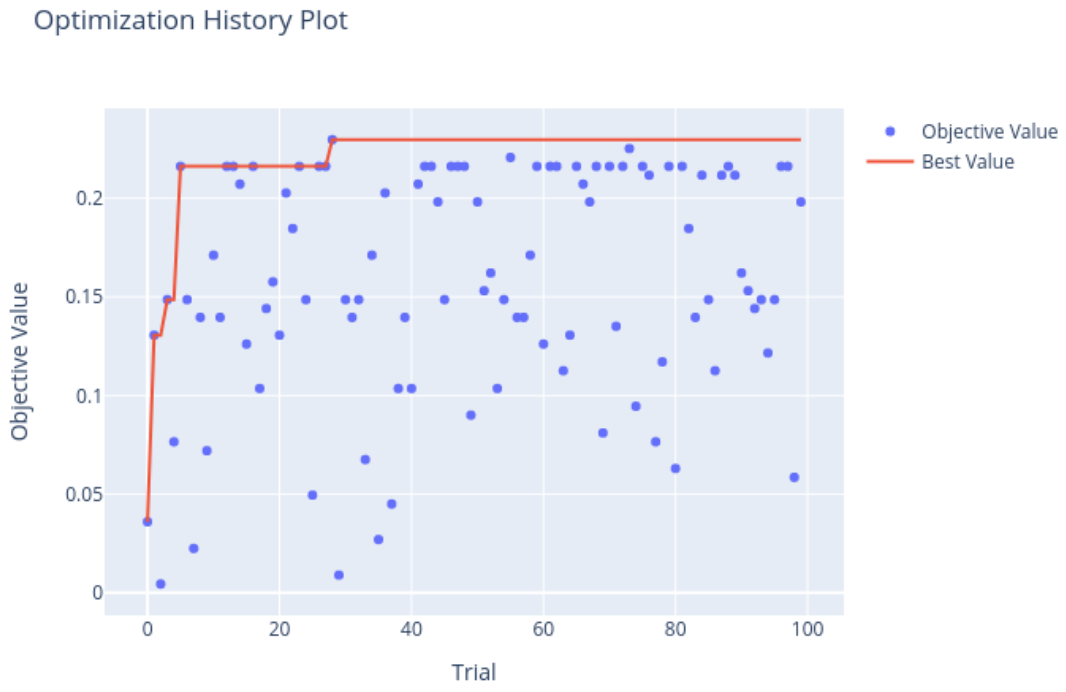
Furthermore, we ran an initial hyperparameter tuning on the RNN model with the feature-based approach (which is our focus as it is expected to perform best) to asses each



hyperparameter and select a set of hyperparameters that will work optimally wit our model. The tuning was implemented using the tuning library *optuna* and 100 different hyperparameter combinations were evaluated.

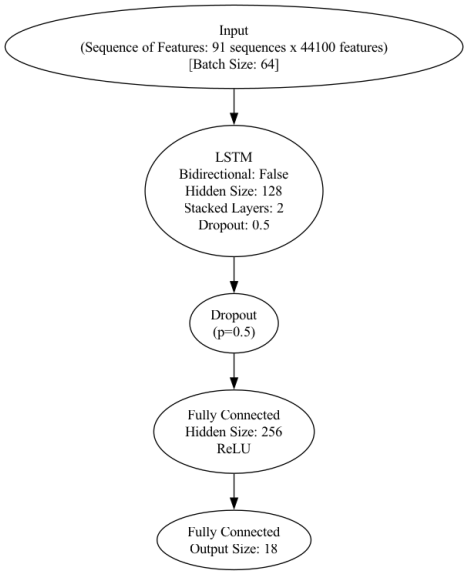
The following plots help as understand the impact each hyperparameter has on the model performance and generalization ability:


Hyperparameter Importances
learning_rate
hidden_layers
hidden_size
dropout_probability
num_layers
cell_type
bidirectional
skip_connections
gradient_clipping
0.53
0.12
0.12
0.09
0.04
0.04
0.03
0.02
0.01
Hyperparameter
0
0.1
0.2
0.3
0.4
0.5
Hyperparameter Importance

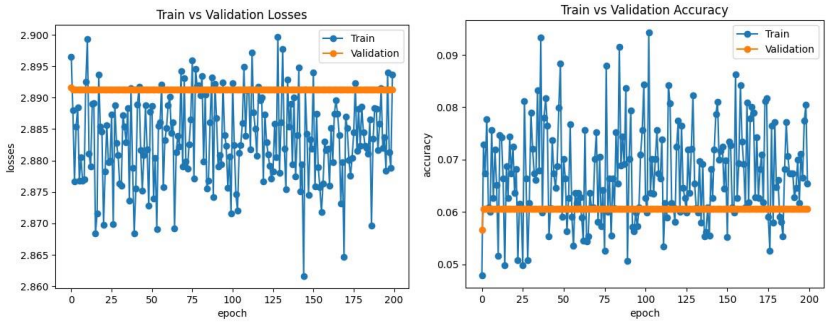

Optimization History Plot
Objective Value
Best Value
0.2
0.15
0.1
0.05
0
Objective Value
0
20
40
60
80
100
Trial

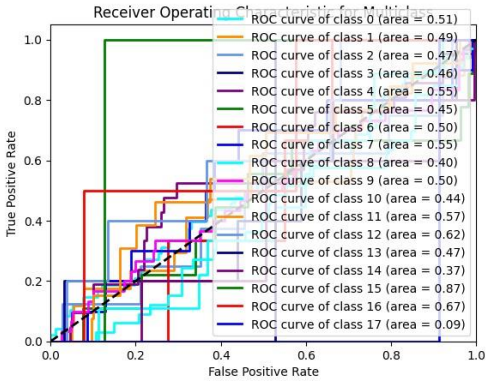
RNN architecture with raw data approach

Since the performance of the model that used raw data was poor in any scenario, we picked a straightforward architecture to demonstrate the performance of the network. The hyperparameters used can be observed in the following figure.



The model was trained for 200 epochs with a batch size of 64. The results were low, as expected, as the amount of training data was not adequate enough for the model to generalize and as a result was significantly underfitted. Indicative of the model's performance are the below performance metrics and evaluation plots.





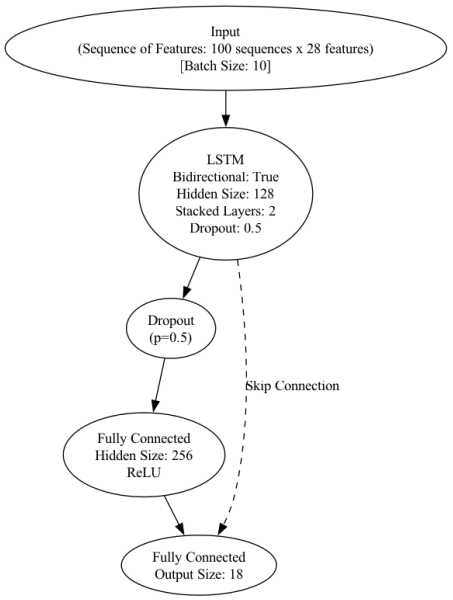

```
-------------------------------------------
Model classification report:
                  precision    recall  f1-score   support

      Aegean-sea       0.15      0.06      0.09        48
       Macedonia       0.14      0.05      0.07        21
      Ionian-sea       0.00      0.00      0.00         9
 Southern-Italy        0.00      0.00      0.00         9
      Asia-minor       0.01      1.00      0.03         2
  Central-Greece       0.00      0.00      0.00         1
         Thessaly      0.00      0.00      0.00        33
     Peloponnese       0.00      0.00      0.00         5
           Epirus      0.00      0.00      0.00         8
 Mainland-Greece       0.00      0.00      0.00         1
           Crete       0.00      0.00      0.00        17
          Pontus       0.00      0.00      0.00        30
          Euboea       0.17      0.10      0.12        10
           Egypt       0.00      0.00      0.00        13
          Thrace       0.00      0.00      0.00         1
         Romania       0.10      0.20      0.13         5
          Cyprus       0.00      0.00      0.00         3
         Kythira       0.00      0.00      0.00         5

        accuracy                           0.04       221
       macro avg       0.03      0.08      0.02       221
    weighted avg       0.06      0.04      0.03       221

-------------------------------------------
Model accuracy:  0.03619909502262444
-------------------------------------------
```

RNN architecture with feature-based approach

Having run the hyperparameter tuning, we chose the following hyperparameters for our model:

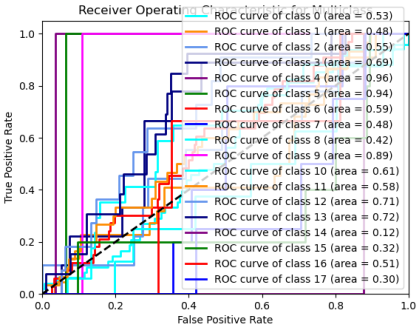
- learning rate: 0.001
- stacked rnn layers: 2
- skip connections: True
- dropout: 0.5
- hidden layers: 2
- gradient clipping: True
- cell type: lstm
- hidden size: 128
- bidirectional: True

The model was training for 100 epochs with a batch size of 10. The final network architecture is the following:



The results were underwhelming as the model could not generalize once again. Despite that, compared to the raw data approach the performance was drastically improved.

Below are some performance metrics:




```
Model classification report:
                  precision    recall  f1-score   support

      Aegean-sea       0.22      1.00      0.36        48
      Asia-minor       0.00      0.00      0.00        22
 Central-Greece        0.00      0.00      0.00         9
           Crete       0.00      0.00      0.00         9
          Cyprus       0.00      0.00      0.00         2
           Egypt       0.00      0.00      0.00         1
          Epirus       0.00      0.00      0.00        33
          Euboea       0.00      0.00      0.00         5
      Ionian-sea       0.00      0.00      0.00         8
         Kythira       0.00      0.00      0.00         1
       Macedonia       0.00      0.00      0.00        17
Mainland-Greece        0.50      0.03      0.06        30
     Peloponnese       0.00      0.00      0.00        11
          Pontus       0.00      0.00      0.00        13
         Romania       0.00      0.00      0.00         1
 Southern-Italy        0.00      0.00      0.00         4
        Thessaly       0.00      0.00      0.00         3
           Thrace       0.00      0.00      0.00         4

        accuracy                           0.22       221
       macro avg       0.04      0.06      0.02       221
    weighted avg       0.12      0.22      0.09       221

Model accuracy:  0.22171945701357465
```

Due to the large class imbalance, the model being underfit and the few test samples, the results are poor.

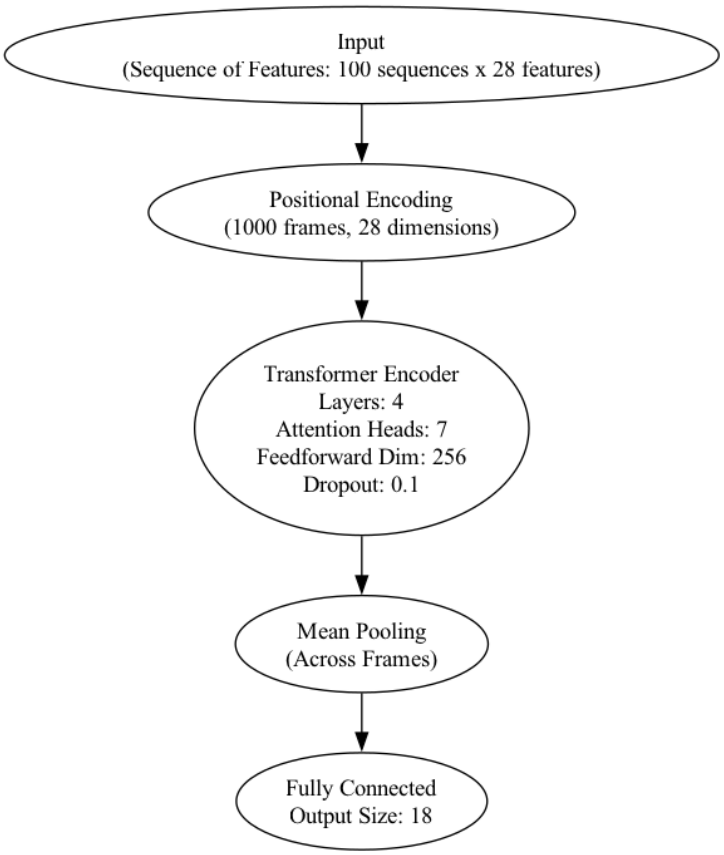
6.4. Transformers

With the recent popularity of attention (the building block of transformers) in deep learning, transformer-based architectures are starting to being used in audio classification. We implemented this architecture to simply demonstrate that a Transformer model is also an option when classifying songs.

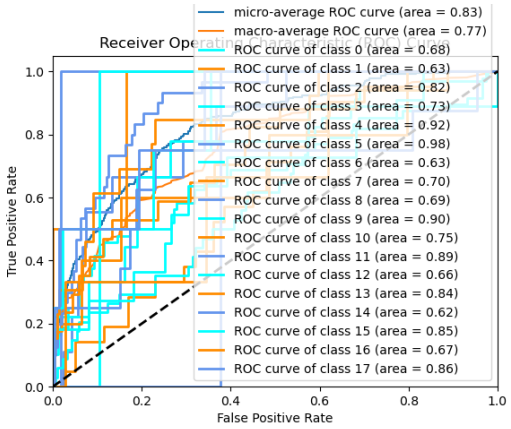
Again, a straightforward architecture is used. We employed a Transformer model with carefully chosen hyperparameters to optimize performance on sequences of audio features. The feature dimension is set to 28, meaning each input frame comprises 28 features, capturing essential characteristics of the audio. We used seven attention heads in the multi-head attention mechanism to allow the model to focus on different parts of the input sequence simultaneously, enhancing its ability to learn diverse patterns (and also 7 is a product of 28 which is a common practice to use attention heads divisible with the feature dimension). The model includes four encoder layers, providing depth to learn complex hierarchical representations of the input data. The feedforward network within each encoder layer has a dimension of 256, enabling the model to perform sophisticated transformations and capture intricate patterns. We also used positional encodings of length 1000 to provide the model with information about the position of each frame in the sequence, which is crucial for maintaining the temporal structure of the audio. A dropout rate of 0.1 is applied to prevent overfitting by introducing regularization during training.

In our Transformer model, positional encodings are crucial for maintaining the temporal order of input sequences. Despite each sequence containing 100 frames of 28 features, we chose a positional encoding length of 1000 to ensure comprehensive coverage of potential sequence lengths encountered during training. This choice allows the model to effectively learn and utilize the temporal relationships between frames, enhancing its ability to capture long-term dependencies and accurately classify audio features based on their sequential context. Positional encodings bridge the gap for Transformer models, ensuring they can interpret and process sequential data like audio with accuracy and efficiency.

Following up is a simple visualization of our network:



The model was trained for 200 epochs with a batch size of 64.



The model managed to outperform its RNN counterpart with the same features.

Model classification report:

	precision	recall	f1-score	support
Aegean-sea	0.38	0.54	0.44	48
Asia-minor	0.00	0.00	0.00	21
Central-Greece	0.00	0.00	0.00	9
Crete	1.00	0.22	0.36	9
Cyprus	0.00	0.00	0.00	2
Egypt	0.00	0.00	0.00	1
Epirus	0.30	0.21	0.25	34
Euboea	0.00	0.00	0.00	5
Ionian-sea	0.00	0.00	0.00	8
Kythira	0.00	0.00	0.00	1
Macedonia	0.33	0.18	0.23	17
Mainland-Greece	0.38	0.90	0.53	30
Peloponnese	0.40	0.18	0.25	11
Pontus	0.28	0.62	0.38	13
Romania	0.00	0.00	0.00	1
Southern-Italy	1.00	0.25	0.40	4
Thessaly	0.00	0.00	0.00	3
Thrace	0.00	0.00	0.00	4
accuracy			0.34	221
macro avg	0.23	0.17	0.16	221
weighted avg	0.30	0.34	0.28	221

7. Comparative Analysis

7.1. Performance Comparison

Given the task of multiclass classification, accuracy is a crucial metric for us, as it reflects our ability to effectively cover and distinguish between all classes. Below is a comparison of the various approaches we have implemented:

Architecture	SVM (feat)	FFNN (feat)	CNN (Spec)	CNN (Mel)	RNN (raw)	RNN (feat)	Transformer (feat)
Accuracy	0.38	0.23	0.17	0.33	0.04	0.22	0.34
Avg Training Time	23s	110s	~1h	~1h	2h	10m	5m

As we observed, the baseline implementations tend to outperform our more complex architectures. This result is unexpected, as increasing the complexity of the model typically leads to better performance. It's important to consider the variety of data and features used for each model, including raw data, Librosa features, and PyAudio analysis features.

7.2. Result Analysis

This outcome could be due to several reasons:

Underfitting: The results indicate that our models are heavily underfitted and require more training instances to properly generalize across all classes. The limited amount of data for each class, particularly the minority classes, restricts the models' ability to learn the intricate patterns needed for accurate classification.

Class Imbalance: The significant imbalance between classes means that the models are biased towards the dominant classes, leading to poor overall performance. When focusing only on the dominant classes, which have the majority of training examples, our models can achieve accuracy levels of around 60%. This suggests that with more balanced and comprehensive training data, the models' performance could improve substantially.

Feature Representation: The choice of features plays a crucial role in the models' performance. While we experimented with raw data, Librosa features, and `PyAudioAnalysis`

features, it is possible that the features were not adequately capturing the necessary information for effective classification. Exploring more advanced feature extraction techniques or using a combination of different features could potentially yield better results.

Model Complexity: While more complex architectures have the potential for better performance, they also require more data and computational resources to train effectively. Our current dataset and computational setup might not be sufficient to fully leverage the capabilities of these complex models. Simplifying the models or optimizing their training process might help in achieving better results with the available resources.

8. Conclusions and Future Work

8.1. Summary of Findings

In conclusion, we experimented with various deep learning architectures to evaluate their performance and behavior in classifying regions of songs. Our findings indicate that using spectrograms as input features and Convolutional Neural Networks (CNNs) as the primary model architecture provides a solid foundation for audio classification tasks. Spectrograms effectively capture the time-frequency representation of audio signals, making them suitable for identifying distinct audio patterns.

Furthermore, we observed that more complex sequential models, such as transformers, also show significant potential in delivering accurate results. Transformers, with their attention mechanisms, can effectively capture long-range dependencies in audio data, which is particularly useful for understanding the context and nuances in music.

8.2. Challenges and Limitations

The multiclass classification task is inherently challenging due to the large number of classes and the significant class imbalance. In our specific application, we faced the added complexity of dealing with audio features that are not fully representative of the intricate characteristics present in the data. The extracted features primarily capture basic information sufficient to distinguish different genres of music, where there is a high degree of contrast in audio characteristics. However, our problem requires classifying different regions of songs that belong to the same or similar genres. These regions often share many common features, such as tones, language, and instruments, making them difficult for a model to distinguish.

Additionally, the presence of noise in our data, such as clapping, crowd talking, and other background sounds, further complicates the classification process. This noise introduces variability that can obscure the subtle differences between classes, making accurate classification even more challenging.

The handling of large-scale datasets and the implementation of complex model architectures present significant time and computational challenges. These challenges are exacerbated by the fact that some aspects of the processing cannot be parallelized effectively, leading to longer training times and increased computational resource demands.

One of the most critical issues we faced is the inadequacy of training examples for all classes. The limited amount of training data restricts our ability to train a deep learning model that can accurately distinguish between the different classes. This problem is compounded by the fact that the number of sequences produced is insufficient for training sequential models effectively. Sequential models, which are often required for capturing the temporal dynamics in audio data, need a large amount of diverse training sequences to generalize well.

8.3. Future Work

The high-class imbalance means that the model tends to be biased towards the majority classes, often neglecting the minority classes. This imbalance makes it harder to achieve a balanced performance across all classes, leading to poor generalization in real-world scenarios.

To address these issues, we should initially focus on finding and annotating a significantly larger number of training examples from each region. It is also crucial to include many clean and consistent studio recordings that provide clear audio samples. Additionally, we could employ data augmentation techniques, such as time stretching, pitch shifting, and adding synthetic noise, to increase the diversity and robustness of our training data. Scaling the data appropriately can also help in balancing the dataset and improving model performance.

Exploring more advanced feature extraction methods is another critical step. Furthermore, employing transfer learning by using pre-trained models on large audio datasets can help improve the performance when training data is limited. Techniques like fine-tuning a pre-trained model on our specific dataset can leverage the learned features and adapt them to our task.

In addition to these, implementing robust noise reduction and signal processing techniques can help mitigate the impact of background noise. Using algorithms designed to filter out unwanted noise can improve the clarity of the audio samples, making it easier for the model to focus on the relevant features.

In addition to CNNs and transformers, it is worth exploring hybrid models that combine the strengths of both architectures. For instance, using CNNs to extract spatial features from spectrograms and then feeding these features into a transformer model could leverage the advantages of both approaches, potentially leading to even better performance.

Finally, addressing class imbalance through techniques like oversampling the minority classes, under sampling the majority classes, or using cost-sensitive learning methods can help ensure a

balanced performance across all classes. Employing ensemble methods or hybrid models that combine different types of classifiers can also improve robustness and accuracy.

9. References

[1] C. Papaioannou, I. Valiantzas, T. Giannakopoulos, M. Kaliakatsos-Papakostas and A. Potamianos, "A Dataset for Greek Traditional and Folk Music: Lyra", in Proc. of the 23rd Int. Society for Music Information Retrieval Conf., Bengaluru, India, 2022.

[2] Shawn Hershey, Sourish Chaudhuri, Daniel P. W. Ellis, Jort F. Gemmeke, Aren Jansen, R. Channing Moore, Manoj Plakal, Devin Platt, Rif A. Saurous, Bryan Seybold, Malcolm Slaney, Ron J. Weiss, Kevin Wilson, "CNN Architectures for Large-Scale Audio Classification", 29 Sep 2016.

[3] Fabien Brulport, RNN Sound classification

[4] K. Zaman, M. Sah, C. Direkoglu and M. Unoki, "A Survey of Audio Classification Using Deep Learning," in IEEE Access, vol. 11, pp. 106620-106649, 2023, doi: 10.1109/ACCESS.2023.3318015.

[5] Theodoros Giannakopoulos, Aggelos Pikrakis, Chapter 5 - Audio Classification, Introduction to Audio Analysis, Academic Press, 2014, Pages 107-151, ISBN 9780080993881

[6] Theodoros Giannakopoulos, Aggelos Pikrakis, Chapter 6 - Audio Segmentation, Introduction to Audio Analysis, Academic Press, 2014, Pages 107-151, ISBN 9780080993881

[7] Dalibor Mitrović, Matthias Zeppelzauer, Christian Breiteneder, Chapter 3 - Features for Content-Based Audio Retrieval, Advances in Computers, Elsevier, Volume 78, 2010, Pages 71-150, ISSN 0065-2458, ISBN 9780123810199

[8] Audio Deep Learning Made Simple: Sound Classification, Step-by-Step

10. Related Work

- Tsoulou, Kalliopi, 2019, Feature-based Machine Learning Techniques towards Greek Folk Music Classification, Available at: http://hdl.handle.net/11544/29475
- Χριστοδούλου Αννα Μαρία, Computational Analysis of Greek folk music of the Aegean islands, 2022