Audio Classification on the FSDKaggle2018 dataset using Tensorflow,Keras

Anurag Ravi Nimonkar

May 3, 2024

Abstract

This project explores audio classification leveraging representation learning within the TensorFlow framework. The methodology focuses on the initial engineering of wave features derived from raw audio data, which are subsequently used to train convolutional neural networks (CNNs) for effective representation learning. By transforming the audio signals into a structured format amenable to convolutional processing, our system is designed to capture the intrinsic properties and patterns embedded in the sound waves. The feature engineering process is detailed where various envelope features such as the homomorphic envelogram, hilbert envelogram and wavelet decompositions help in extracting meaningful information from the raw audio signals. These engineered features provide a robust foundation for the subsequent layers of convolutional networks. The CNNs are meticulously architected to learn hierarchical representations, effectively capturing both low-level and high-level audio characteristics.This study attempts to reinforces the significance of tailored feature engineering in deep learning and demonstrate an effective audio classification pipeline using representation learning and open up new avenues for research in audio signal processing and machine learning.

1 Introduction

The objective of this experiment is to explore efficacy of engineered signal features for an audio classification task in a 2-step experiment using 1-D Convolutions to learn representations. CNNs are preferred for audio classification tasks since they are able to capture local frequency dependencies and learn robust representations over other alternatives like a Hidden Markov Model(HMM) and Recurrent Networks typically not suited for non-speech data and used for segmentation tasks instead. The learning experiments explore digital signal processing of audio signals and the analysis of results and engineering features requires understanding of audio normalization, interpolation, wave theory, digital filters and sample frequency. To model design, architecture, implementation and analysis of performance requires understanding representation learning, convolutional networks, batch processing, hyper parameter tuning, vector processing and data

analysis. The fundamental goal of this project remains to demonstrate an audio classification pipeline for a 2 step learning experiment using engineered features and learned representations and analysis.

1.1 Feature Engineering

Feature engineering from raw audio involves extracting meaningful and informative characteristics from audio signals to improve the performance of classification models. This process typically starts with the raw audio waveform, which is then transformed into a format that better represents the underlying properties of the sound for analytical purposes.

1.1.1 Homomorphic Envelope

The Homomorphic envelope is a technique often used to separate the source and the filter characteristics of a sound. It usually involves taking the logarithm of the signal's spectrum, performing operations (such as filtering), and then taking the inverse transform. This can help in situations where you need to analyze the underlying spectral envelope of a sound, which is useful for speaker or phoneme recognition in speech analysis. The negative values in the homomorphic envelope indicate that the envelope does not show absolute values but relative changes in the signal's spectral envelope. The choice of pass band (5Hz to 40Hz) is because it's broad enough to detect slow changes in dynamic range across various sounds without being too specific to any particular type of sound. For audio files with rhythmic content, such as musical instruments (e.g., bass drum, tambourine) and environmental noises (e.g., applause, bus), this range helps in detecting the rate of energy changes which relate to tempo or the occurrence of events. This passband is a compromise designed to provide useful information across various types of audio signals without specializing too much in any one attribute like pitch or timbre. It focuses on the overall "energy" shifts within the audio which can be crucial for several forms of analysis, including detecting presence, changes in sound levels, and basic rhythmic patterns. This makes it generally useful for a variety of applications, from simple sound detection to more complex analysis involving rhythms and dynamics.

The homomorphic envelope of a signal s(t) is often used to extract the envelope of the signal in a way that separates the contributions of the rapidly varying fine structure and the slowly varying amplitude envelope. The process involves several steps, usually starting with taking the natural logarithm of the absolute value of the signal's analytic representation (obtained via the Hilbert transform) to linearize convoluted relationships in the frequency domain. Mathematically, it can be defined as:

$\mathbf{E}_{hom(t)} = exp(Hlog(|s(t)|))$, where H is the Hilbert transform

1.1.2 Hilbert Envelope

The Hilbert envelope image illustrates the amplitude envelope of the audio signal. It provides a smooth curve representing the instantaneous amplitude of the audio waveform. The Hilbert envelope is useful for analyzing the amplitude modulations and can be used for detecting the presence of transient sounds or the rhythmic structure of the audio. This envelope starts high and tapers off, which corresponds to the decaying sound indicated in the waveform. The amplitude envelope is a smooth curve that outlines the extremes of an audio waveform and can highlight the temporal structure of the sound events in this dataset, which can be more informative than the frequency-based features alone. The Hilbert envelope of a signal provides a measure of the signal's amplitude envelope, capturing the local amplitude variations. It is defined using the Hilbert transform H(s(t)) of the signal s(t). The envelope is then given by:

$\mathrm{E}_{hil(t)} = \sqrt{(s(t)^2 + H(s(t))^2)}$

1.1.3 Wavelet Decomposition

The Wavelet transform is a powerful tool for time-frequency analysis, capable of providing time and frequency information simultaneously, and is particularly adept at identifying transient features and anomalies in a signal. The Wavelet transform plot shows how the frequency content of the signal changes over time, with the x-axis representing the wavelet coefficient index and the y-axis the coefficient values. Unlike the Fourier transform, wavelet transform can give a better picture of non-stationary signals such as audio, as it can capture both high-frequency events with fine temporal resolution and low-frequency events with fine frequency resolution. Wavelets provide a time-frequency representation of the signal and are especially good at analyzing non-stationary signals because they can capture both high-frequency events (like transients) and low-frequency events (like sustained notes) with high resolution. This makes wavelet features particularly well-suited to a dataset with a wide variety of sounds, as they can capture the nuances of both quick, sharp sounds (like a "gunshot" or "gunfire") and slower, evolving sounds (like "acoustic guitar" or "cello") Wavelet decomposition involves breaking down a signal into a series of wavelets or small waves, which can vary in frequency and position, allowing for multi-resolution analysis of the signal. The decomposition of a signal can be represented as

$\mathrm{s(t)} = \sum_{j,k} c_{j,k}\psi_{j,k}(t)$, where $\psi_{j,k}(t)$ is the scaled and transformed wavelet function and $c_{j,k}$ are the wavelet coefficients.

1.2 Representation Learning

Representation learning in audio classification using TensorFlow involves training models to automatically discover the representations needed for classification directly from raw audio data or pre-processed audio features. The goal is to transform raw audio or engineered features into a form where classification tasks, such as identifying spoken words or music genres, become more manageable.

TensorFlow facilitates this by providing a robust and scalable platform for building and training deep learning models, such as convolutional neural networks (CNNs).

2 Related Work

Almost all deep learning tasks in signal processing follow a conventional approach wherein domain expertise is required to first engineer relevant features and then further processing is a complete black box. Recent work has been trying to mitigate the domain knowledge required to build such a pipeline by shifting towards end-to-end models which are seeing success in segmentation tasks[1]. Using logarithmic audio scaling as in mel-spectrograms and MFCC features has also seen success in classification and segmentation tasks[2]. The paper by Juhan Nam, Keunwoo Choi, Jongpil Lee, Szu-Yu Chou, and Yi-Hsuan Yang[3] also illustrates a sample-level CNN amongst other industry approaches on audio classification, sentiment tagging and music genre classification. This project aims to re-emphasize the foundational knowledge in signal processing and wave theory required to truly fine-tune an audio processing pipeline such as the one used in this experiment.

3 Experiment

The classification pipeline can be broken down into steps - data pre-processing, feature extraction, model training and testing.

3.1 Data Preparation

3.1.1 Dataset

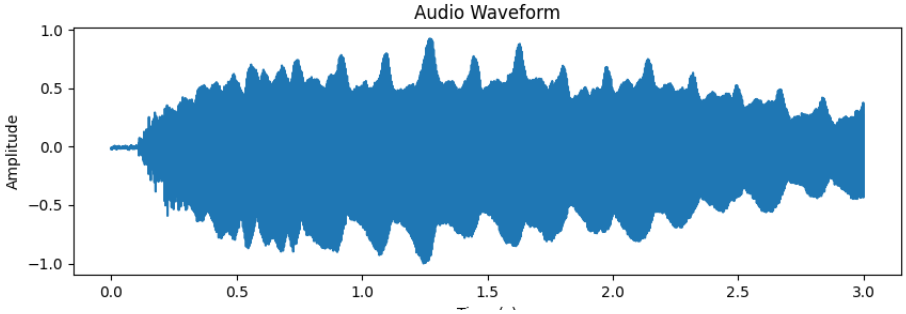
The FSDKaggle 2018 audio dataset has 11,073 audio files annotated with 41 labels - "Acoustic guitar", "Applause", "Bark", "Bass drum", "Burping or eructation", "Bus", "Cello", "Chime", "Clarinet", "Computer keyboard", "Cough", "Cowbell", "Double bass", "Drawer open or close", "Electric piano", "Fart", "Finger snapping", "Fireworks", "Flute", "Glockenspiel", "Gong", "Gunshot or gunfire", "Harmonica", "Hi-hat", "Keys jangling", "Knock", "Laughter", "Meow", "Microwave oven", "Oboe", "Saxophone", "Scissors", "Shatter", "Snare drum", "Squeak", "Tambourine", "Tearing", "Telephone", "Trumpet", "Violin or fiddle", "Writing". The minimum number of audio samples per category in the train set is 94, and the maximum is 300. The duration of the audio samples ranges from 300ms to 30s and all audio samples in this dataset have a single label. The sample rate for the files is 44,100 Hz(high resolution).

3.1.2 Pre-processing

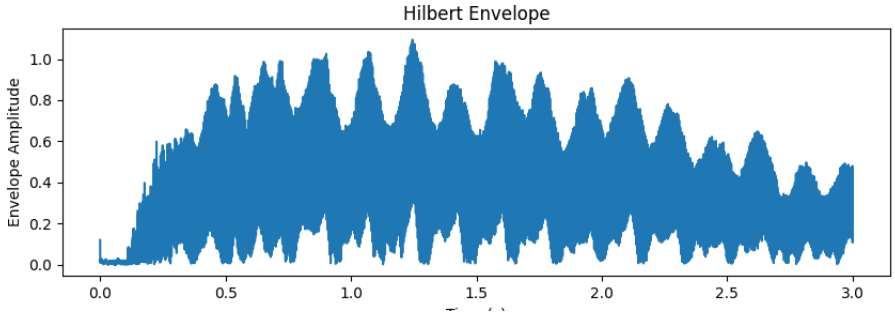
The audio data is loaded and immediately normalized early in the pipeline by feature scaling using min-max normalization y = y/np.max(np.abs(y)) The

normalized signal is then broken into chunks of size = length*sample rate for further processing. Samples shorter than the fixed chunk length(in seconds) are 0 padded to make up the difference. Only the manually verified samples are considered for training. The final training set is reduced and balanced. There are 50 examples from each of the 41 classes reducing the final training set to 2050 samples. The audio is not downsampled from the original 44,100Hz sampling rate. Downsampling is not preferred, the tradeoff being degradation of information in the signal with lesser sample rates. However, some downsampling can help compromise the loss in quality of the data with quantity; allowing processing a larger dataset with a more minimal GPU throughput. These considerations were taken into account and the decision to retain the original sampling rate was made, considering the goals aligned with that of an academic end semester project; processing the whole dataset was not prioritized in this experiment.

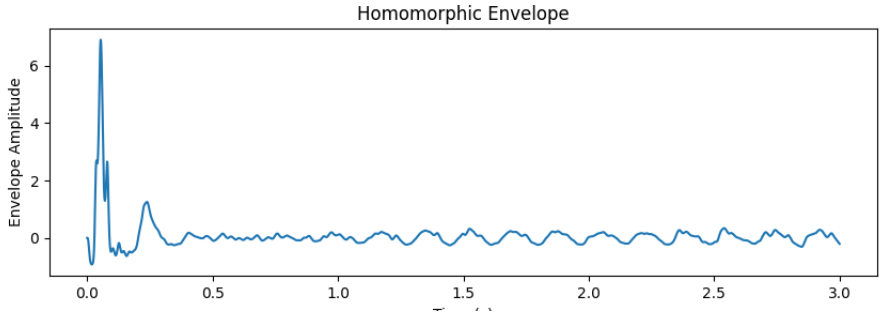
3.2 Feature Extraction



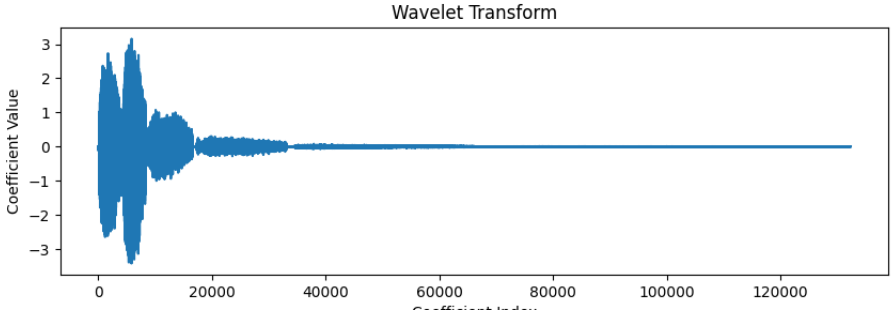
a. original audio chunk



b. hilbert envelope



c. homomorphic envelope



d. wavelet transform

a. Original Audio The chunk is taken from a sample belonging to the class "Cello". The length of the feature arrays are concatenated to match the length of the chunk (the last few samples of the wavelet transform are lost as a result)

b. Hilbert Envelope Cellos produce sound through the vibration of strings caused by bowing or plucking, leading to a smooth variation in amplitude as the note swells and decays. This gradual increase and decrease in volume can be clearly seen in the Hilbert envelope, where peaks correspond to the bowing action causing maximum displacement of the string, and the troughs reflect the points where the string returns to a less excited state. This envelope shows the dynamics of each note played on the cello, highlighting the expressive volume control a cellist has.

c. Homomorphic Envelope The specific bandpass filtering used to generate the homomorphic envelope (5Hz to 40Hz) targets very slow oscillations in the signal's amplitude, often corresponding to the tempo or the rhythmic pacing of a musical piece. For a cello, this could capture the pacing of bowing motions or phrasing in a solo performance, which often involves deliberate and measured changes in intensity and speed. The homomorphic envelope in this case smooths out the fine textural details and focuses on these broad dynamics, emphasizing

how a cellist may sustain notes and use silences effectively, contributing to the distinct emotive quality of the cello.

d. Wavelet Transform The wavelet transform using a Daubechies 4 wavelet is particularly adept at capturing both transient details and localized frequency information. For a cello, which has a rich harmonic content and a distinctive timbre, the wavelet transform can reveal start and end of notes, visible as sharp changes in the wavelet coefficients, Variations in playing technique, such as vibrato or changes in bowing pressure, reflected in modulations of the wavelet coefficients and the rich overtones and the fundamental frequency of notes, which are visible across different scales of the wavelet decomposition. The lower scales (higher frequencies) capture the detailed texture of the sound, including the scratch of the bow on the string, while the higher scales (lower frequencies) can show the fundamental tone of the note being played.

Each of these features—Hilbert envelope, homomorphic envelope, and wavelet transform—provides a unique lens through which the distinctive sound of a cello can be analyzed. By comparing these features against known characteristics of cello sounds (like those from acoustic studies or sound libraries), one can convincingly argue that the audio recording is indeed of a cello, based on the captured envelopes and wavelet features that align well with the expected acoustic behavior of cello sounds.

3.3 Model Training

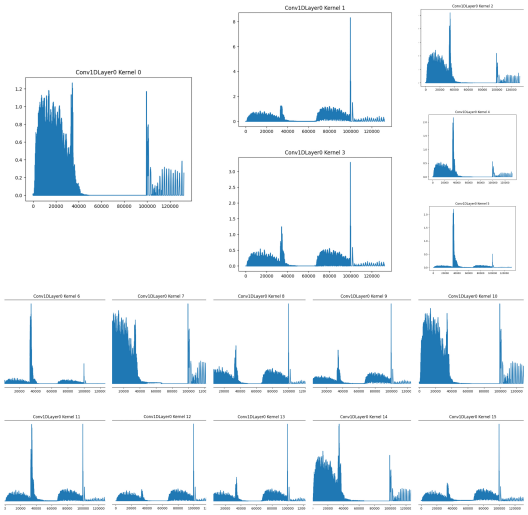
```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv1d (Conv1D)             (None, 132298, 16)        64

 max_pooling1d (MaxPooling1  (None, 66149, 16)         0
 D)

 conv1d_1 (Conv1D)           (None, 66147, 32)         1568

 max_pooling1d_1 (MaxPoolin  (None, 33073, 32)         0
 g1D)

 conv1d_2 (Conv1D)           (None, 33071, 64)         6208

 max_pooling1d_2 (MaxPoolin  (None, 16535, 64)         0
 g1D)

 conv1d_3 (Conv1D)           (None, 16533, 64)         12352

 max_pooling1d_3 (MaxPoolin  (None, 8266, 64)          0
 g1D)

 flatten (Flatten)           (None, 529024)            0

 dense (Dense)               (None, 16)                8464400

 dropout (Dropout)           (None, 16)                0

 dense_1 (Dense)             (None, 41)                697

=================================================================
Total params: 8485289 (32.37 MB)
Trainable params: 8485289 (32.37 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

e. model

3.3.1 Model Hyper-parameters

Kernel size = 3
MaxPooling, pool size = 2
Optimizer = “Adam”, learning rate = 0.0001
Dropout rate = 0.2
Length = 3 seconds
Batch size = 128
Input shape = ((sample rate*length),1)
Epochs = 50
Xtrain.shape = (batch size, sample rate*length, of features), Ytrain.shape = (batch size,)

3.3.2 Activations



f. activations for first convolution

The images above show the activations/representations learned by the first convolution in my model for the aforementioned audio waveform, which belongs to the class "Cello". Each kernel captures different aspects of the audio waveform, translating these into a feature map that the neural network uses to understand and process the audio data. By analyzing the activation patterns in relation to the waveform, we can infer what each part of the network is focusing on, such as detecting the onset of sounds, capturing frequency changes, or identifying rhythmic patterns. This helps in refining the model for more precise audio analysis tasks, like distinguishing different types of sounds(audio classification/tagging) or understanding speech dynamics. Cellos produce rich, resonant tones that can vary in pitch, timbre, and intensity, and have distinctive attack, sustain, and decay phases in their notes. The audio waveform shows the following characteristics that can be related to a cello:

Onset Of Notes : The beginning of a Cello note is characterized by a distinct attack where the bow strikes the strings, leading to an initial spike in amplitude.

Sustain and Vibrato : As the note is held, the Cello exhibits a sustained sound with possible vibrato, which would appear as oscillations in amplitude within the waveform.

Changes in Pitch or Dynamics : Changes in pitch or dynamics during playing would lead to corresponding variations in the waveform's amplitude and frequency.

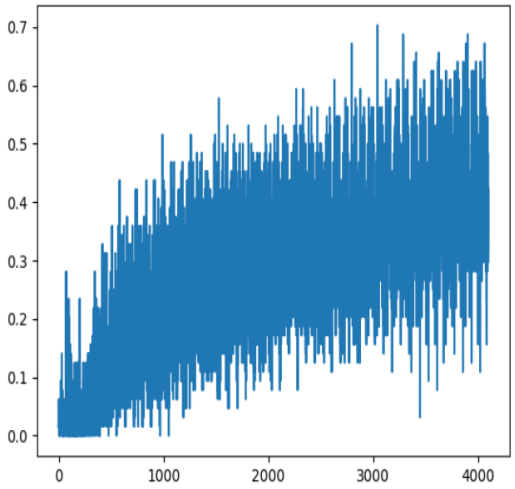
Kernels 0 and 7 (High Initial Response) : Likely sensitive to the onset and attack phase of the Cello notes. The initial response tapering off could reflect the decay of the attack into the smoother sustain phase of the note.

Kernels 1, 2, 4, 5, 6 (Sharp Isolated Spikes): These kernels may be capturing specific transient features of the Cello, such as the striking of the bow on the strings, changes in bowing pressure, or quick changes in pitch. The sharp spikes could correspond to points where the bow changes direction or when the player shifts positions on the fingerboard.

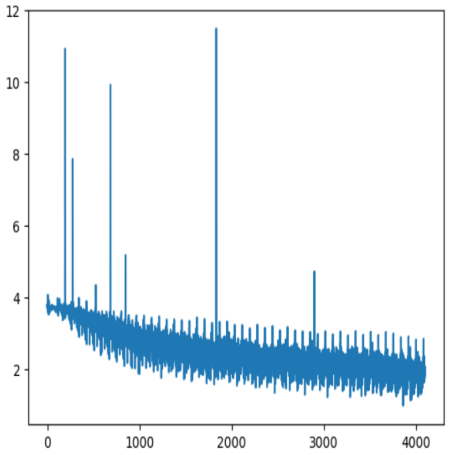
Kernels 3, 8 and 9 (Multiple Peaks): Could be responding to rhythmic patterns in the Cello playing, or repeated motifs and phrases within the piece. These kernels might also be picking up on the vibrato or the repetitive bowing technique common in Cello playing.

Cellos produce a range of frequencies, typically lower than many other string instruments but with significant harmonic content. Kernels showing responses across a broad range might be capturing these harmonic elements, essential for timbre recognition. The kernels displaying gradual activations might be sensitive to the dynamic range of the Cello, capturing the crescendos and decrescendos typical in classical Cello compositions.

3.4 Results



g. training accuracy



h. training loss

```
Precision: 0.05811561612642704
Recall: 0.11049723756906077
F1 Score: 0.06758721637861748
```

i. testing metrics

training loss The training loss shows multiple sharp peaks which may indicate occasional large errors in prediction during training. These peaks decrease in frequency and magnitude as training progresses, indicating some improvement in the model's learning. The general trend is a decrease in loss, suggesting that the model is learning to minimize the error over iterations. However, the existence of peaks suggests instability which may be due to insufficient outlier handling.

training accuracy The training accuracy starts very low and increases gradually, showing a positive trend but with significant fluctuation. Again, the general trend is improvement implying the model is somewhat learning to classify correctly. The noticeable fluctuations could imply that the model struggles with a consistently reliable prediction across different stages of training. This can be influenced by factors like model complexity, noise in data, or overfitting on specific parts of the training data.

precision Precision measures the accuracy of positive predictions. It is calculated as the ratio of true positive (correctly predicted positive instances) to the sum of true positive and false positive (incorrectly predicted positive instances). In this case, the precision is quite low at approximately 0.058.

recall measures the ability of the classifier to find all the positive samples. It is calculated as the ratio of true positive to the sum of true positive and false negative (positive instances incorrectly classified as negative). The recall score here is around 0.110, also relatively low.

F1 score The F1 score is the harmonic mean of precision and recall. It provides a balance between precision and recall. The F1 score here is approximately 0.068, indicating that the model's performance is low on both precision and recall.

Overall, the model is generalizing very poorly. I believe a possible fix to the problem is using more of the dataset in a balanced fashion like in this experiment for a more complex model architecture however this solution is restricted by system and GPU requirements. Similarly, validation logic would make the script too bulky since we are training in batches and we extract features for every audio chunk. The training matrix is a large np.ndarray and the model is complex, making space for an additional validation matrix would compromise data set size and model complexity. Hence, validation metrics are unavailable.

4 References

1. Fernando, Tharindu Ghaemmaghami, Houman Denman, Simon Sridharan, Sridha Hussain, Nayyar Fookes, Clinton. (2019). Heart Sound Segmentation Using Bidirectional LSTMs With Attention. IEEE Journal of Biomedical and Health Informatics. PP. 1-1. 10.1109/JBHI.2019.2949516.
2. Gourisaria, M.K., Agrawal, R., Sahni, M. et al. Comparative analysis of audio classification with MFCC and STFT features using machine learning techniques. Discov Internet Things 4, 1 (2024). https://doi.org/10.1007/s43926-023-00049-y
3. Nam, Juhan Choi, Keunwoo Lee, Jongpil Chou, Szu-Yu Yang, yi-hsuan. (2019). Deep Learning for Audio-Based Music Classification and Tagging: Teaching Computers to Distinguish Rock from Bach. IEEE Signal Processing Magazine. 36. 41-51. 10.1109/MSP.2018.2874383.
4. My work in digital signal processing(DSP) as a machine learning intern at AI Health Highway is the reference for : feature extraction and data pre-processing steps.
5. My course-work in COMP47700 (Speech And Audio) and COMP47650 (Deep Learning) is the reference for : data analysis, audio theory,representation learning using convolutions and model architecture.
6. My experience in sound engineering(mixing and mastering) my own music is reference for : music theory, linear filtering, noise reduction.