Maqam Classification of Quranic Recitations using Deep Learning Report

Faisal Omari[a], Mayas Ghantous[b], Nimrod Peleg[c]

[a]B.Sc. Computer science student University of Haifa Computer Science Department
[b]B.Sc. Computer science student University of Haifa Computer Science Department
[c]Faculty member at the Department of Electrical Engineering Technion-Israel Institute of Technology

Abstract

The maqam is a musical mode or scale that characterizes the melodic style of Quranic recitation. The maqam classification of Quranic recitations is a challenging task that requires a high level of skill and knowledge. In this paper, we present a unique approach for maqam classification using deep learning. We use two datasets of audio samples from different reciters and maqamat, and extract various features from them, such as mel-frequency cepstral coefficients (MFCC), chroma, root mean square (RMS) energy, zero-crossing rate (ZCR), spectral roll-off, spectral centroid, and spectral bandwidth. We train different models, such as artificial neural networks (ANNs), convolutional neural networks (CNNs), and long short-term memory networks (LSTMs), to predict the maqam labels from the audio features. We evaluate our models using different metrics, such as accuracy, loss, and confusion matrices. We achieve promising results, with an accuracy of 94.3% for the best model. Our work establishes the superiority and effectiveness of using deep learning for maqam classification of Quranic recitations and offers a valuable tool for researchers and practitioners in this field.

1. Introduction

The Quran is the sacred scripture of Islam, revealed to the Prophet Muhammad (pbuh) over a span of 23 years. The Quran is recited by Muslims in various situations, such as prayers, ceremonies, and festivals. The Quranic recitation is not merely a verbal act, but also a musical and artistic one, as it follows certain rules and styles that enhance the beauty and eloquence of the words of Allah. One of the key aspects of the Quranic recitation is the maqam, which is a musical mode or scale that characterizes the melodic style of the reciter. The maqam is not fixed or predetermined, but rather chosen by the reciter according to his preference, mood, or context. The maqam can also change within a single recitation, depending on the meaning and emotion of the verses. The maqam is not only a musical element, but also a spiritual and cultural one, as it reflects the reciter's personality, background, and identity.

The maqam classification of Quranic recitations is a difficult task that requires a high level of skill and knowledge. The maqamat are not well-defined or standardized, but rather subjective and flexible. The maqamat are also affected by various factors, such as the reciter's voice, accent, style, speed, and intonation. Moreover, the maqamat are not exclusive or independent, but rather overlapping and interrelated. There are many maqamat that are used for Quranic recitation, but the most common ones are eight: Ajam, Bayat, Hijaz, Kurd, Nahawand, Rast, Saba, and Seka. Each maqam has its own characteristics, such as pitch range, intervals, melodic patterns, and mood.

The maqam classification of Quranic recitations has many applications and benefits for both researchers and practitioners in this field. For researchers, it can help them analyze and understand the musical and linguistic features of the Quranic recitation, and explore the relationship between the maqam and the meaning of the verses. For practitioners, it can help them learn and improve their recitation skills, and choose the suitable maqam for different occasions and contexts. Moreover, it can help them appreciate and enjoy the diversity and beauty of the Quranic recitation.

In this paper, we propose a new approach for maqam classification using deep learning. We will train some deep learning models on two datasets, that we've collected as part of the project, we will extract a set of features for each audio sample that's related to the classification problem and can help in the classifying process.

We also suggest some possible ways to improve our work and achieve better results for the maqam classification task.

2. Background

2.1. Quran

Quran: The Quran is the holy book of Islam, believed to be the word of God revealed to Prophet Muhammad through the angel Gabriel. It consists of 114 chapters, or surahs, that cover various topics such as faith, morality, history, law, and guidance. The Quran is recited in Arabic and has been translated into many languages [3].

2.2. Mujawad

Mujawwad is a term that describes a melodic style of Quran recitation that is known throughout the Muslim world. Mujawwad is derived from the root word j-w-d, which means to improve, beautify, or perfect. Mujawwad is a form of artistic expression that adds emotion, variation, and embellishment to the recitation. Mujawwad recitation follows certain musical modes or scales, known as maqamat (plural of maqam) in Arabic. There are eight popular maqamat that are used for Quran recitation: Ajam, Bayat, Hijaz, Kurd, Nahawand, Rast, Saba, and Seka [2] [1] [4].

2.3. Tartel

Tarteel is the term that describes the recitation of the Quran at a moderate pace, with proper order and without haste. Tarteel is derived from the root word r-t-l, which means to arrange, measure, or regulate. Tarteel is considered an obligatory manner of recitation for every Muslim who recites the Quran, as it ensures the clarity and understanding of the divine message. Tarteel also reflects the reverence and devotion of the reciter to the words of Allah. Allah says in the Quran: "And recite the Quran with tarteel." [25:32] [21], the used datasets in this project consisted of tarteel recitations.

2.4. Pitches

Pitches are the perceived frequencies of musical sounds, which can be measured in hertz (Hz). In Western music, pitches are usually divided into 12 equal steps per octave, forming the chromatic scale. However, in some Eastern music traditions, such as Arabic, Turkish, and Persian music, pitches can be subdivided into smaller intervals called quarter tones or microtones, which are half the size of a semitone. These pitches allow for more expressive and nuanced melodies and modulations [29].

2.5. Maqamat

Maqamat are a system of melodic modes or scales used in Arabic music and other related traditions. Each maqam has a characteristic pattern of intervals, a starting note or tonic, and a set of typical melodic phrases or motifs, the most popular maqamat are 8, Ajam, Bayat, Hijaz, Kurd, Nahawand, Rast, Saba, and Seka. Maqamat can be modulated or transposed to different pitches and can vary in performance according to the context, mood, and style of the musician. Maqamat are similar to but distinct from the raga system of Indian music and the makam system of Turkish music [15].

2.6. Musical Notes

Musical notes are symbols that represent musical sounds in written form. Notes can indicate the pitch, duration, and articulation of a sound, as well as other aspects such as dynamics, tempo, and expression. Notes are usually written on a staff or stave, which consists of five horizontal lines and four spaces. Different clefs are used to indicate the range of pitches on the staff. Notes can also be modified by accidentals, which alter their pitch by a semitone or half-step [23].

2.6.1. MFCC

MFCC stands for mel-frequency cepstral coefficients, which are features that represent the short-term power spectrum of a sound. MFCCs are derived from applying a Fourier transform, a mel-scale filter bank, and a discrete cosine transform to a sound signal. MFCCs are widely used in speech recognition, speaker identification, and music analysis [6]. The following figures show the MFCC of each maqam [figure: 1-8].

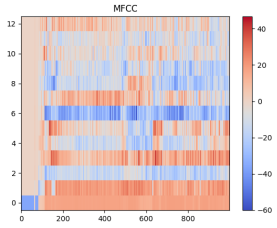

Figure 1: Ajam MFCC

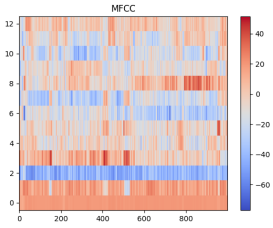

Figure 2: Bayat MFCC

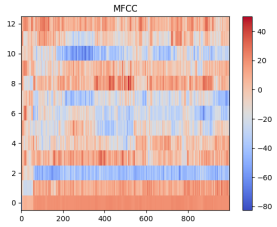

Figure 3: Hijaz MFCC

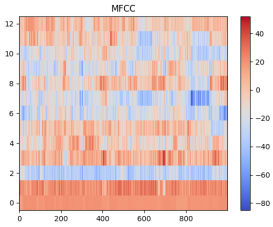

Figure 4: Kurd MFCC

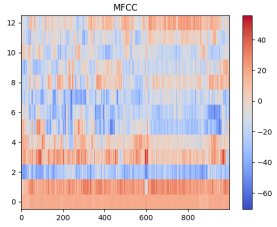

Figure 5: Nahawand MFCC

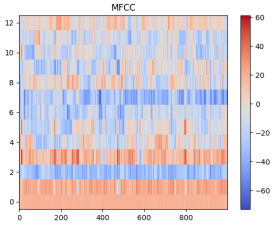

Figure 6: Rast MFCC

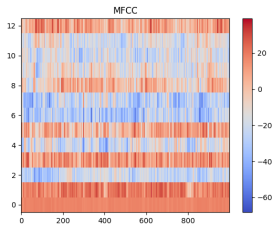

Figure 7: Saba MFCC

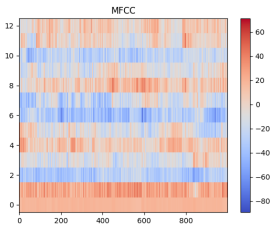

Figure 8: Seka MFCC

MFCC extraction process (figure 9):

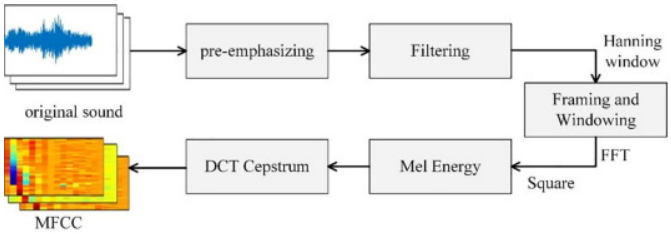

Figure 9: MFCC extraction

2.6.2. Chorma

Chroma is a feature that represents the pitch content of a sound or music signal. Chroma is obtained by mapping the frequency spectrum of a sound to 12 pitch classes that correspond to the notes of the chromatic scale. Chroma can capture the harmonic and melodic characteristics of a sound or music signal, regardless of its timbre or octave [7]. The following figures show the Chroma of each maqam [figure: 10-17].

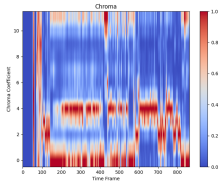

Figure 10: Ajam Chroma

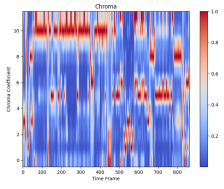

Figure 11: Bayat Chroma

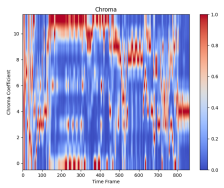

Figure 12: Hijaz Chroma

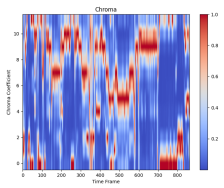

Figure 13: Kurd Chroma

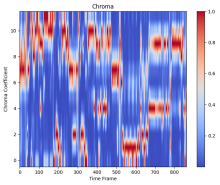

Figure 14: Nahawand Chroma

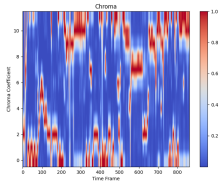

Figure 15: Rast Chroma

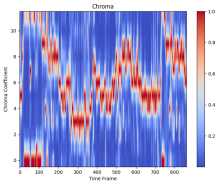

Figure 16: Saba Chroma

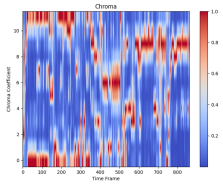

Figure 17: Seka Chroma

2.6.3. RMS

RMS stands for root mean square, which is a measure of the magnitude or amplitude of a signal. RMS is calculated by squaring the values of a signal, taking the average of the squares, and then taking the square root of the average. RMS can be used to estimate the loudness or energy of a sound or music signal [18]. The following figures show the RMS of each maqam [figure: 18-25].

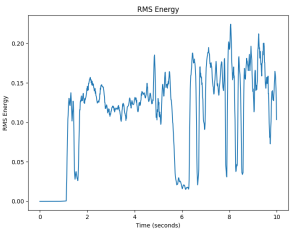

Figure 18: Ajam RMS

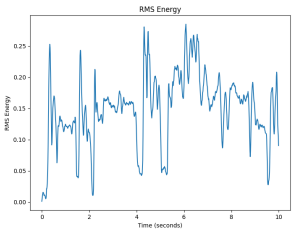

Figure 19: Bayat RMS

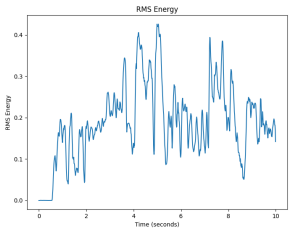

Figure 20: Hijaz RMS

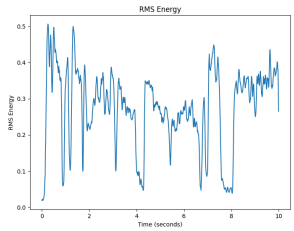

Figure 21: Kurd RMS

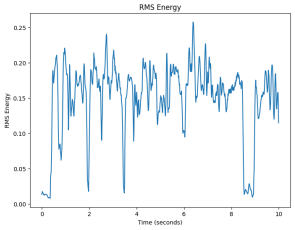

Figure 22: Nahawand RMS

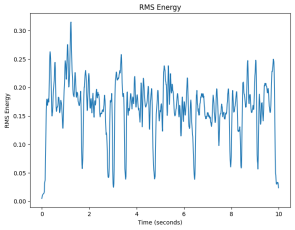

Figure 23: Rast RMS

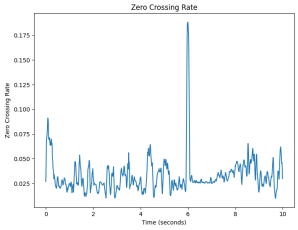

Figure 30: Nahawand ZRC

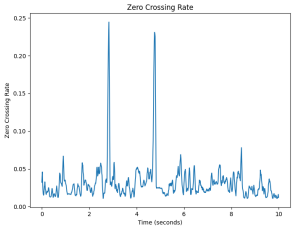

Figure 31: Rast ZRC

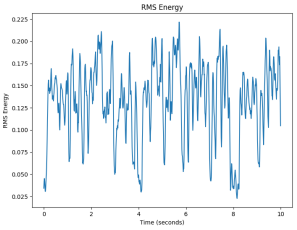

Figure 24: Saba RMS

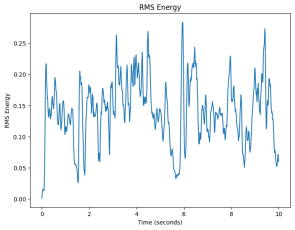

Figure 25: Seka RMS

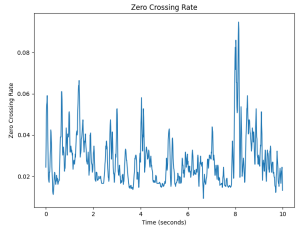

Figure 32: Saba ZRC

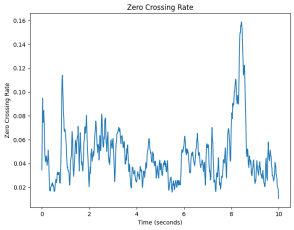

Figure 33: Seka ZRC

2.6.4. ZRC

ZRC stands for zero-crossing rate, which is a measure of the rate at which a signal changes sign from positive to negative or vice versa. ZRC is calculated by counting the number of times that a signal crosses the zero level within a given time frame and dividing it by the length of the frame. ZRC can be used to estimate the frequency or pitch of a sound or music signal, especially for percussive or noisy sounds [9]. The following figures show the ZRC of each maqam [figure: 26-33].

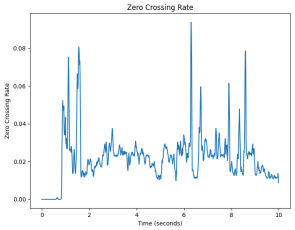

Figure 26: Ajam ZRC

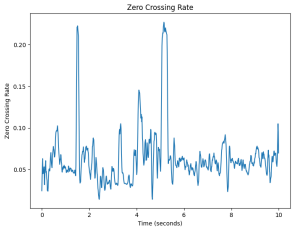

Figure 27: Bayat ZRC

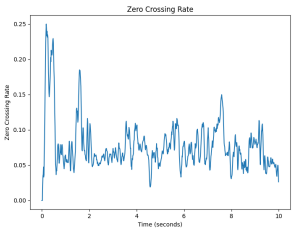

Figure 28: Hijaz ZRC

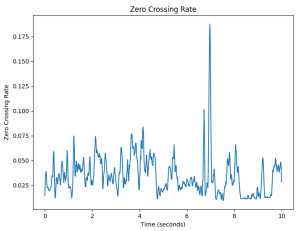

Figure 29: Kurd ZRC

2.6.5. Spectral Centorid

Spectral centroid is a feature that represents the center of mass or gravity of the frequency spectrum of a sound or music signal. Spectral centroid is calculated by multiplying each frequency bin by its magnitude and dividing the sum by the total magnitude. Spectral centroid can be used to estimate the brightness or timbre of a sound or music signal, as higher values indicate more high-frequency content and lower values indicate more low-frequency content [20].
Figure 34 shows the Spectral Centroid of arbitrarily chosen maqam:

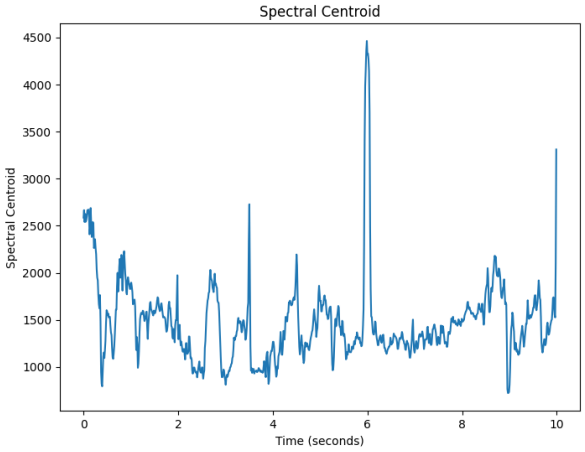

Figure 34: Nahawand Spectral Centroid

2.6.6. Spectral Roll-off

Spectral roll-off is a feature that represents the frequency below which a certain percentage (usually 85 or 95 percent) of the total energy of the frequency spectrum of a sound or music signal is contained. Spectral roll-off is calculated by finding the frequency bin that satisfies this condition and interpolating its value if necessary. Spectral roll-off can be used to estimate the bandwidth or range of frequencies of a sound or music signal, as higher values indicate more high-frequency content and lower values indicate more low-frequency content [30].
Figure 35 shows the Spectral Roll-off of arbitrarily chosen maqam:

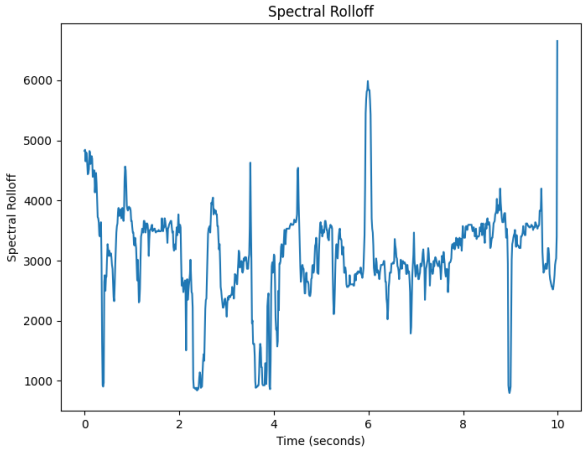

Figure 35: Nahawand Spectral Roll-Off

2.6.7. Spectral bandwidth

Spectral bandwidth is a feature that represents the spread or variation of the frequency spectrum of a sound or music signal around its mean value. Spectral bandwidth is calculated by taking the square root of the second central moment or variance of the frequency spectrum weighted by its magnitude. Spectral bandwidth can be used to estimate the smoothness or roughness of a sound or music signal, as higher values indicate more variation and lower values indicate more concentration [14].
Figure 36 shows the Spectral bandwidth of arbitrarily chosen maqam:

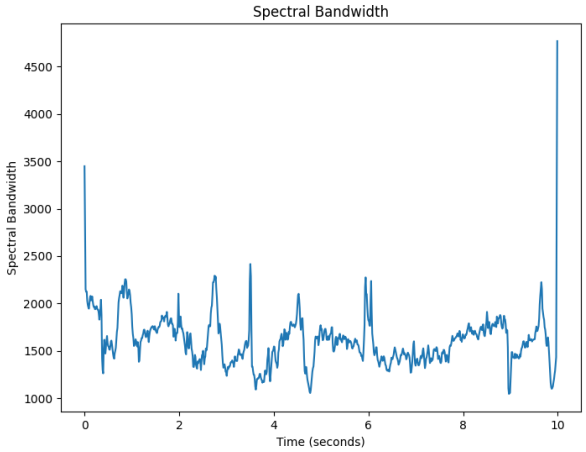

Figure 36: Nahawand Spectral Bandwidth

2.7. Deep learning

Deep learning is a branch of machine learning that uses artificial neural networks with multiple layers to learn from data. Deep learning can perform complex tasks such as image recognition, natural language processing, speech synthesis, and reinforcement learning, often achieving state-of-the-art results. Deep learning models can learn from large amounts of unlabeled or unstructured data and extract high-level features and representations from them [8].

2.8. Features

Deep learning features are learned representations or abstractions of data that are extracted by deep learning models. Features can be low-level or high-level depending on their position in the network hierarchy. Low-level features capture basic patterns or edges in the data, while high-level features capture more complex concepts or semantics in the data. Features can be used for various purposes such as classification, clustering, generation, or transfer learning [13].

2.9. Neural networks

Neural networks are computational models that are inspired by the structure and function of biological neurons and synapses in the brain. Neural networks consist of layers of interconnected nodes or units that process and transmit information. Each node has an activation function that determines its output based on its inputs and weights. Neural networks can learn from data by adjusting their weights using various learning algorithms such as gradient descent or backpropagation. Neural networks can perform tasks such as regression, classification, clustering, or dimensionality reduction [10]. Figure 37 shows Neural Network architecture:

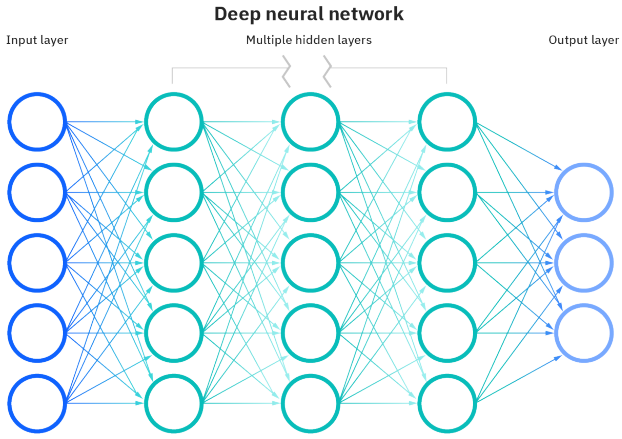

Figure 37: Deep Neural Network architecture

2.10. CNN

CNN stands for convolutional neural network, which is a type of neural network that is specialized for processing data that has a grid-like structure such as images or videos. CNNs use convolutional layers that apply filters to local regions of the input data to extract features. CNNs also use pooling layers that reduce the spatial dimensions of the data by applying a function such as max, average, or sum. CNNs can perform tasks such as object detection, face recognition, semantic segmentation, or style transfer [32].

2.11. LSTM

LSTM stands for long short-term memory, which is a type of recurrent neural network (RNN) that can handle long-term dependencies in sequential data. LSTM networks use memory cells that store and update information over time. LSTM networks also use gates that control the flow of information into and out of the memory cells. LSTM networks can perform tasks such as natural language processing, machine translation, speech recognition, or time series forecasting [22].

2.12. ANN

ANN stands for artificial neural network, which is a general term for any neural network model that is based on artificial neurons or nodes. ANN models can have different architectures, such as feedforward, recurrent, convolutional, or generative. ANN models can also have different learning methods, such as supervised, unsupervised, or reinforcement. ANN models can perform various tasks depending on their design and application domain [11].

2.13. Nashaz/ Cocophona

Nashaz (Arabic) or cocophona (Greek) are terms that describe the dissonance or unpleasantness caused by combining two or more sounds that are not harmonious or compatible with each other according to certain musical rules or conventions. Nashaz or cocophona can result from using pitches that are out of tune, intervals that are not consonant, or maqamat that are not appropriate for the context or mood. Nashaz or cocophona can be intentional or unintentional, and can have different effects on the listener depending on their musical background and preference [12].

2.14. audio

Audio is a term that refers to any sound or signal that can be heard by humans or machines. Audio can be analog or digital, mono or stereo, and have different formats and qualities depending on the source, medium, and device. Audio can be used for various purposes such as communication, entertainment, education, or research [24].

2.15. sample rate

Sample rate is a measure of how often a continuous signal is sampled or measured per unit of time, usually in hertz (Hz) or samples per second. Sample rate determines the highest frequency that can be captured or reproduced by a signal without distortion, according to the Nyquist-Shannon sampling theorem. Sample rate also affects the size and quality of a digital audio file, as higher sample rates require more storage space and processing power but offer more fidelity and resolution [28].

2.16. wav

WAV is a file format for storing uncompressed digital audio data. WAV files use the RIFF (Resource Interchange File Format) container format and can store audio data in various formats such as PCM (Pulse-Code Modulation), ADPCM (Adaptive Differential Pulse-Code Modulation), or IEEE floating point. WAV files are widely supported by various software and hardware platforms and offer high quality and flexibility, but also take up a lot of storage space [5].

2.17. Python

Python is a high-level, interpreted, general-purpose programming language that supports multiple paradigms such as object-oriented, imperative, functional, and procedural. Python is known for its simple and elegant syntax, dynamic typing, rich standard library, and large community of developers and users. Python is widely used for various applications such as web development, data analysis, machine learning, scientific computing, and scripting [31].

2.18. PyTorch

PyTorch is an open-source framework for machine learning and deep learning based on the Torch library. PyTorch provides a flexible and expressive way of defining and executing computational graphs using tensors and autograd. PyTorch also offers various tools and libraries for data loading, preprocessing,

visualization, debugging, and distributed training. PyTorch is popular among researchers and practitioners for its ease of use, speed, and scalability [19].

3. Problems

The problem of maqam classification consists of several challenging subproblems, with few existing solutions. Our research identified only one published solution in the article "Classifying Maqams of Qur'anic Recitations Using Deep Learning" by Shahriar [27]. This limited availability of solutions posed a significant challenge for our project. To address this, we divided the problem into the following subproblems for a more systematic approach:

3.1. Dataset

Initially, we faced the challenge of obtaining an appropriate dataset for training, one that could yield promising results and high accuracy. While we started with the Maqam478 dataset [25], we soon realized its limitations. To construct a more suitable dataset, we had to overcome various obstacles, including sourcing and filtering Quranic recitations in MP3 format. Ensuring that the audio data was accurately classified for specific maqams, free from outliers, was achieved with the expertise of Faisal Omari, who demonstrated a 100% accuracy in maqam differentiation. Additionally, we encountered issues related to the quality of some recitations, including low sample rates and audio quality, leading us to filter out many samples.

3.2. Feature Selection

Initially, we attempted to train the model directly on the audio signal, which proved ineffective in achieving desirable accuracy levels. This led us to the critical task of selecting the most suitable feature for model training. After extensive research, we decided to use the features mentioned in the background section, with MFCC emerging as the most promising choice. MFCC is a common feature used in audio data analysis, such as speech recognition, and has a track record of producing highly accurate models and research results.

While MFCC stood out as the primary feature, we also experimented with several other features, the results of which are discussed in the "Training Approaches" section. During this process, we encountered issues with model performance, particularly when using CNNs. We concluded that the small dataset was a limiting factor. As suggested by our supervisor, Mr. Nimrod, we pivoted temporarily to a different approach – conducting tournaments between randomly chosen pairs of maqams to qualify which pair achieved the highest accuracy. This demonstrated the potential for the problem to be solved with a larger dataset. However, we also managed to achieve remarkable accuracy exceeding 85% through persistent effort, prompting us to revert to our original project path.

3.3. Model Selection

Selecting an appropriate model posed another challenge, as it required significant experimentation, training, and parameter tuning. In this regard, we explored three models: ANN, CNN, and LSTM, and in this article and report, we present the top-performing models from our experimentation.

4. Coding

Upon the recommendation of Mr. Ibraheem, our project supervisor, we chose to utilize PyTorch instead of TensorFlow for our project. This decision was based on his experience, and we proceeded accordingly. Our implementation was carried out using Python in conjunction with the PyTorch library.

For the computational infrastructure, we relied on Faisal's local PC, which, while not ideal for deep learning tasks, featured the following hardware specifications: - CPU: Intel Core i5-9400F - GPU: NVIDIA GeForce GTX 1060 Ti 6GB

Despite the hardware limitations, we approached the project with professionalism and expertise. We carefully selected parameters such as batch size and model architecture to ensure that the training process could effectively run on the given hardware, and we successfully achieved this balance.

To bolster our knowledge and skills in deep learning, we completed an entire course based on Mr. Nimrod's notes. This course, CS231N, focuses on deep learning for computer vision and is available as a playlist on YouTube. Over the course of the first two months, as documented in our monthly reports on GitHub, we diligently worked through the course material, completing exercises and assignments.

Throughout the project, we made extensive use of Python libraries, as previously mentioned in the background section. For data preprocessing, we employed the soundfile and librosa libraries.

While the actual coding phase did not consume a significant amount of time, we did encounter initial challenges when running the code. These difficulties arose because we dove directly into the training process. We faced issues with PyTorch and CUDA drivers, prompting Faisal to transition to a Linux operating system to address and resolve these problems. In our GitHub repository, we provided comprehensive explanations and comments for the entire codebase. Additionally, we made an effort to organize the code in an efficient and professional manner, utilizing Jupyter Notebook and creating separate Python files with classes for each model.

5. What We Have Learned

Throughout the course of our project, we have gained valuable insights and knowledge that have deepened our understanding of several key aspects related to maqam classification using deep learning. Here, we summarize the key lessons and takeaways from our journey:

5.1. Complexity of Maqam Classification

The problem of maqam classification proved to be multifaceted and challenging. We discovered that it is a problem with limited existing solutions, and addressing it required a systematic approach. Our project allowed us to appreciate the intricacies involved in classifying maqams accurately, from data collection to feature selection and model training.

5.2. Data Collection and Preprocessing

One of the initial hurdles we faced was the lack of readily available datasets tailored for our specific task. To tackle this, we learned the importance of constructing a suitable dataset and the challenges associated with filtering and organizing Quranic recitations for accurate maqam classification. Our collaboration with experts in the field, like Faisal Omari, also taught us the significance of data quality in achieving reliable results.

5.3. Feature Selection

Selecting appropriate features for training our models was a crucial step in our project. We delved into the world of audio feature extraction and found that MFCC emerged as a powerful choice. This experience emphasized the significance of feature engineering in building effective deep learning models.

5.4. Model Selection and Training

Choosing the right model architecture and training process was another key lesson. We explored different deep learning models, including ANN, CNN, and LSTM, and gained insights into their strengths and weaknesses for maqam classification. Our efforts in fine-tuning model parameters highlighted the importance of experimentation and iteration in achieving optimal results.

5.5. Hardware and Infrastructure Challenges

Working with hardware limitations, particularly a CPU-GPU combination that was not ideal for deep learning, taught us the importance of resource optimization and efficient parameter tuning. Our transition to a Linux operating system to address compatibility issues underscored the need for adaptability and problem-solving in the face of technical challenges.

5.6. Educational Growth

Engaging with Mr. Nimrod's comprehensive CS231N course on deep learning for computer vision provided us with a strong foundation in deep learning concepts and techniques. Completing the course's exercises and assignments deepened our knowledge and skill set, equipping us for the challenges of our project.

In summary, our journey in tackling the maqam classification problem has been a transformative learning experience. We have developed expertise in data collection, preprocessing, feature engineering, model selection, and efficient resource utilization. Our commitment to continuous learning and problem-solving has been instrumental in achieving success and advancing our understanding of deep learning in the context of maqam classification.

6. Dataset

One of the challenges of maqam classification of Qur'anic recitations is the lack of publicly available datasets that are labeled with the maqam labels. In this work, we present a novel dataset of Qur'anic recitations labeled with one of the eight popular maqāmāt.

6.1. Dataset Description:

A dataset is a collection of data that is used for machine learning purposes. A dataset can be divided into input variables (features) and output variables (targets). The features are the data that the machine learning model takes as input, and the targets are the data that the model tries to predict.

A good dataset should have the following characteristics:

- It should be relevant to the problem domain and the task at hand.
- It should be large enough to capture the complexity and variability of the data distribution.
- It should be balanced and representative of the different classes or categories of the target variable.
- It should be clean and free of errors, outliers, missing values, or noise.
- It should be well-annotated and labeled with meaningful tags or metadata.

A bad dataset can have the opposite characteristics:

- It can be irrelevant or unrelated to the problem domain or the task at hand.
- It can be too small or too large to fit or process with the available resources.
- It can be imbalanced or skewed towards some classes or categories of the target variable.
- It can be dirty or corrupted with errors, outliers, missing values, or noise.
- It can be poorly annotated or labeled with ambiguous or incorrect tags or metadata.

Extracting features from a dataset is the process of transforming the raw data into a more suitable format for machine learning. Features can be extracted by applying various techniques such as:

- Data cleaning: Removing or correcting errors, outliers, missing values, or noise from the data.
- Data transformation: Changing the scale, type, or distribution of the data to meet the requirements of the machine learning algorithm.
- Data reduction: Reducing the dimensionality or complexity of the data by selecting or combining relevant features.

Extracting features from a dataset can help to improve the quality and performance of machine learning models by:

- Enhancing the signal-to-noise ratio of the data.
- Reducing the computational cost and time of training and testing.
- Increasing the accuracy and generalization ability of the models.
- Facilitating the interpretation and explanation of the models.

6.2. Data Collection:

In this section, we describe how we collected and preprocessed our data for the maqam classification task. We started our data collection from an existing dataset, named "Maqam-478" [25] [26], which was introduced by Shahriar and Tariq [27]. This dataset consists of 478 audio samples from 2 different reciters, each reciting a verse from the Qur'an using a specific maqam. The dataset covers the eight popular maqāmāt (plural of maqam) that we aim to classify in this work. However, we noticed that the Maqam-478 dataset had some limitations, such as:

- It only included two reciters, namely Bandar Balila and Mohammad Ayoub, who performed all the eight maqāmāt.
- It should be large enough to capture the complexity and variability of the data distribution.
- It had some imbalanced classes, where some maqāmāt had more samples than others.

To overcome these limitations and enhance the results of our model, we decided to collect more data from additional sources. We searched for online platforms that provided Qur'anic recitations from various reciters and downloaded audio files that matched our criteria. We selected six more reciters who performed all or most of the eight maqāmāt. The reciters are:

1. Bandar Balila
2. Mohammad Ayoub
3. Yasser Dosri
4. Faisal Omari
5. Saed al Ghamdi
6. Ahmad al Nufis
7. Mushari al Afasy
8. Raed al Kurdi

We manually annotated each audio file with the corresponding maqam label.

6.3. Data Preprocessing

For data preprocessing, we applied the following steps to both the Maqam-478 dataset [25], [26] and our newly collected data:

- We trimmed all the audio samples to 30 seconds to make them equal in size and suitable for our model input.
- We resampled all the audio files to 44100 Hz to make them consistent in sampling rate and quality.
- We iterated over the data and checked it carefully for outliers or errors. We removed or corrected any samples that had nashaz (discordance) or did not match the intended maqam label.
- We balanced the data by ensuring that each maqam had a similar number of samples across different reciters.

As a result of our data collection and preprocessing, we obtained two datasets that we used for our experiments. The first dataset is a smaller one that consists of four reciters who are similar to each other and have almost no outliers or errors. The reciters are:

1. Bandar Balila
2. Mohammad Ayoub
3. Mushari al Afasy
4. Yasser Dosri

This dataset is almost fully balanced and has about 90 samples for each maqam, resulting in a total of 699 samples. This dataset will be referred to as $Dataset_1$ in the following sections. The second dataset is a larger one that consists of all the eight reciters that we collected data from. This dataset is more diverse and challenging, but also more imbalanced and noisy. It has different numbers of samples for each maqam and reciter, resulting in a total of 3869 samples. This dataset will be referred to as $Dataset_2$ in the following sections. [16] [17]

6.4. Data Split:

In this section, we explain how we split our data into training, validation, and test sets. We used the `train_test_split` function from the scikit-learn library in Python, which randomly shuffles and splits the data according to a given ratio. We experimented with different splitting percentages and evaluated the performance of our models on each split. We found that the best split for our data was 70% for training, 15% for validation, and 15% for testing. This split ensured that we had enough data for training our models, while also having enough data for tuning the model parameters and assessing the model generalization. We applied this split to both $Dataset_1$ and $Dataset_2$.

6.5. Feature Extraction:

We extracted the features of the audio samples in our datasets using the Librosa library in Python, which provides various tools and functions for audio analysis and processing. We extracted the following features for each audio sample:

- Mel-frequency cepstral coefficients (MFCC)
- Chroma
- Root mean square (RMS) energy
- Zero crossing rate (ZCR)
- Spectral roll-off
- Spectral centroid
- Spectral bandwidth

6.6. Data Statistics:

Datasets stats are shown in table 1.

Table 1: Descriptive statistics of $dataset_1$ and $dataset_2$

Statistic	$dataset_1$	$dataset_2$
Number of samples	699	3869
Number of reciters	4	8
Class distribution	Ajam: 15.9% Bayati: 12.0% Hijaz: 14.0% Kurd: 10.5% Nahawand: 13.7% Rast: 11.7% Saba: 11.2% Seka: 11.0%	Ajam: 14.0% Bayati: 9.2% Hijaz: 20.6% Kurd: 8.8% Nahawand: 9.0% Rast: 13.9% Saba: 13.4% Seka: 11.1%

7. Training approaches

In the following sections, we will explain our training approach, in some of the models we chose, we trained them over two datasets, the smaller dataset and the larger dataset we mentioned previously, the smaller one will be mentioned as $dataset_1$, and the larger one will be mentioned as $dataset_2$.

7.1. 2D MFCC

Our initial approach to training the model involved utilizing the Mel-frequency cepstral coefficients (MFCC) of each audio sample. As mentioned earlier, MFCC is considered one of the most effective features for tackling this classification problem, a fact that will be demonstrated in the subsequent results. For each sample, we extracted the 2D-shaped MFCC representation. The following models are trained on $dataset_1$, due to the fact that we didn't get the results we hope on them, we moved forward to approaches without training on $dataset_2$

7.1.1. CNN

Considering the 2D nature of MFCC and its analogy to image features, the CNN emerged as the most suitable model for exploiting this feature. As previously discussed, CNNs have proven to be highly effective for processing image data. Given that MFCC can be treated as an image feature for each audio sample (as depicted in the figures presented previously), the CNN was the obvious first choice for our experimentation.

49 shows the model architecture:
Accuracy = 61.09785%

Table 2: CNN 2D-MFCC model architecture

Layer	Input	Output	Dropout	BN	AF
1-CNN	20	64	0.2	64	RELU
2-CNN	64	32	0.2	32	RELU
3-fc	32	512	0.1	512	RELU
4-fc	512	256	0.1	256	RELU
5-fc	256	64	0	64	RELU
output	64	8	0	no	softmax

CNN layers are with kernel=3 and stride=1 and padding=1
Max pool was used after CNN layers, with kernel=1 and stride=2
Hyper parameters:
learning rate = 0.001
number of epochs = 35
batch size = 64
Figure 38 shows training & validation loss.
Figure 39 shows training & validation accuracy.

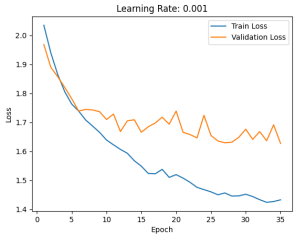

Figure 38: CNN 2D-MFCC loss

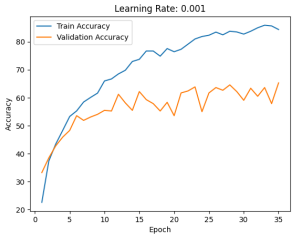

Figure 39: CNN 2D-MFCC accuracy

7.1.2. LSTM

As an alternative to the CNN, we also explored the Long Short-Term Memory (LSTM) model. LSTM is a type of recurrent neural network (RNN) that excels at capturing sequential dependencies in data. While CNNs are well-suited for image-based tasks, LSTM is particularly adept at handling sequential data, making it an excellent choice for analyzing the temporal patterns present in the 2D MFCC representations of our audio samples. By leveraging the sequential memory capabilities of LSTM, we aimed to harness the time-related patterns inherent in the MFCC features to further improve our classification results. Since LSTM considers the entire audio sequence and maintains contextual information over time, it holds promise in effectively capturing the nuances and long-term dependencies present in the audio data.

Table 3 shows the model architecture:
Accuracy = 54.28571%

Table 3: LSTM 2D-MFCC model architecture

Layer	Input	Output	Dropout	AF
1-LSTM	20	1024	0.25	RELU
2-LSTM	1024	512	0.25	RELU
3-LSTM	1024	512	0.25	RELU
4-fc	32	512	0.2	RELU
5-fc	512	256	0.2	RELU
6-fc	256	64	0	RELU
output	64	8	0	softmax

CNN layers are with kernel=3 and stride=1 and padding=1
Max pool was used after CNN layers, with kernel=1 and stride=2
Hyper parameters:
learning rate = 0.001
number of epochs = 40
batch size = 16
Figure 40 shows training & validation loss.
Figure 41 shows training & validation accuracy.

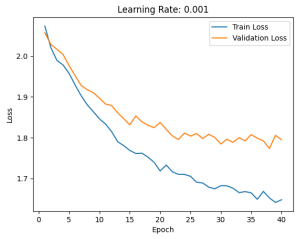

Figure 40: LSTM 2D-MFCC loss

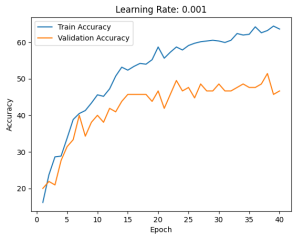

Figure 41: LSTM 2D-MFCC accuracy

7.1.3. ANN

In addition to the previously mentioned models, we tried to use the ANN model on 1D MFCC, and that is by vectorizing the 2D MFCC to make it fit in the ANN model.

Table 4 shows the model architecture:
Accuracy = 74.57142%

Table 4: ANN vectorized MFCC model architecture

Layer	Input	Output	Dropout	BN	AF
1	1D MFCC	1024	0.2	1024	RELU
2	1024	512	0.2	512	RELU
3	512	256	0.1	256	RELU
4	256	128	0.1	128	RELU
5	128	64	0	64	RELU
output	64	8	0	no	softmax

Hyper parameters:
learning rate = 0.0001
number of epochs = 150
batch size = 64
Figure 42 shows training & validation loss.
Figure 43 shows training & validation accuracy.
Figure 44 shows the Confusion matrix.

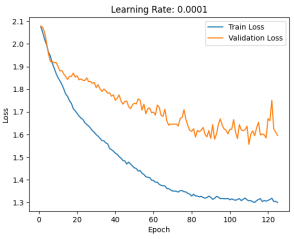

Figure 42: ANN 1D-MFCC loss

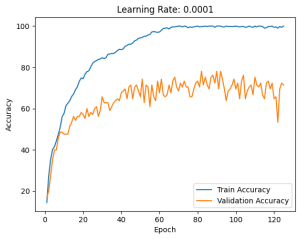

Figure 43: ANN 1D-MFCC accuracy

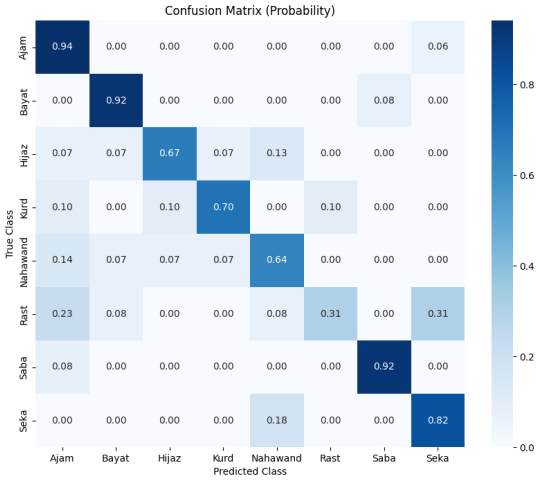

Figure 44: ANN 1D-MFCC confusion Matrix

7.2. Other features

All the models that will be presented in the next few sections are trained using ANN model, due to the fact that all the features are of shape 1D.

7.2.1. MFCC mean

We trained the ANN model on the MFCC coefficient mean on the two datasets.
For $dataset_1$ (The best results in this paper):
Accuracy = 94.29%
Table 5 shows the model architecture:

Table 5: ANN MFCC-means $dataset_1$ model architecture

Layer	Input	Output	Dropout	BN	AF
1-fc	20	1024	0.2	1024	RELU
2-fc	1024	512	0.1	512	RELU
3-fc	512	256	0.1	256	RELU
4-fc	256	128	0	128	RELU
5-fc	128	64	0	64	RELU
output	64	8	0	no	softmax

Hyper parameters:
learning rate = 0.0001
number of epochs = 150
batch size = 64
Figure 45 shows training & validation loss.
Figure 46 shows training & validation accuracy.
Figure 47 shows the ROC Curve.
Figure 48 shows the Confusion matrix.

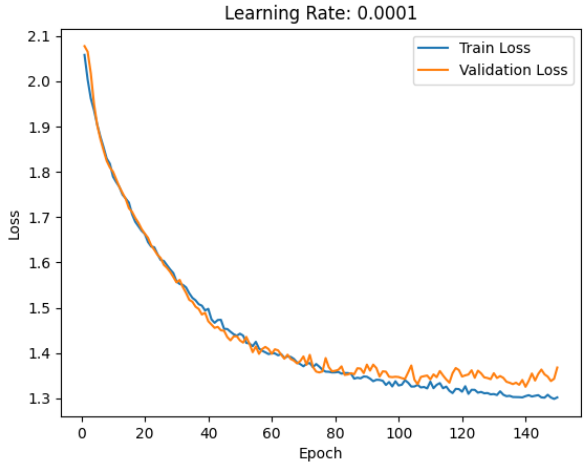

Figure 45: ANN MFCC-means $dataset_1$ loss

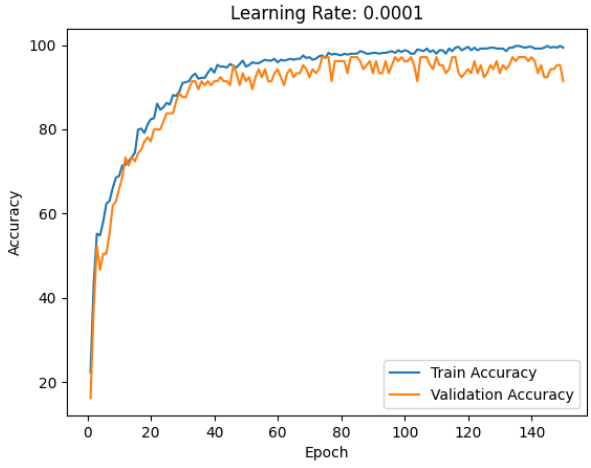

Figure 46: ANN MFCC-means $dataset_1$ accuracy

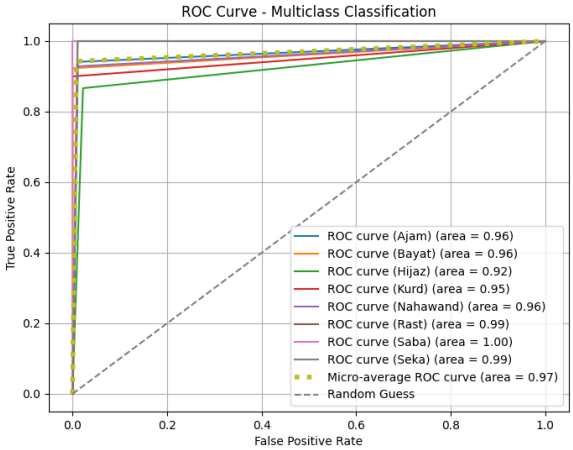

Figure 47: ANN MFCC-means $dataset_1$ ROC

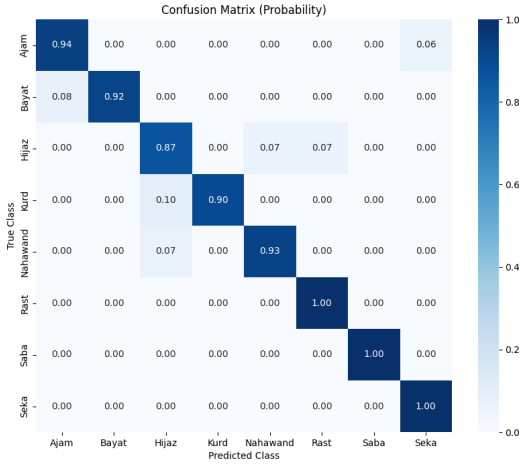

Figure 48: ANN MFCC-means $dataset_1$ confusion Matrix

Upon Mr. Ibraheem's recommendation, we conducted tests using the model on the unmutual portions of dataset1 and dataset2. This allowed us to assess the model's generalization capability beyond the common data shared by both datasets. We achieved an accuracy of 79.6% in this evaluation, indicating the robustness of our model in recognizing maqams in previously unseen data. This result underscores the model's potential for broader applications beyond the initial datasets.

For $dataset_2$:
Accuracy = 85%
Table 6 shows the model architecture:

Table 6: ANN MFCC-means $dataset_2$ model architecture

Layer	Input	Output	Dropout	BN	AF
1-fc	20	4096	0.2	4096	RELU
2-fc	4096	1024	0.2	1024	RELU
3-fc	1024	512	0.1	512	RELU
4-fc	512	256	0.1	256	RELU
5-fc	256	128	0	128	RELU
6-fc	128	64	0	64	RELU
output	64	8	0	no	softmax

Hyper parameters:
learning rate = 0.0001
number of epochs = 150
batch size = 64
Figure 49 shows training & validation loss.
Figure 50 shows training & validation accuracy.
Figure 51 shows the Confusion matrix.

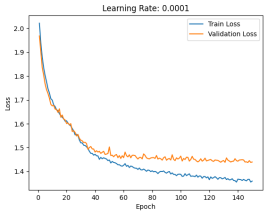

Figure 49: ANN MFCC-means $dataset_2$ loss

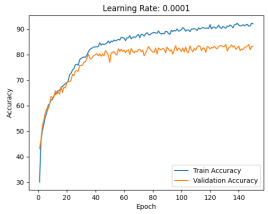

Figure 50: ANN MFCC-means $dataset_2$ accuracy

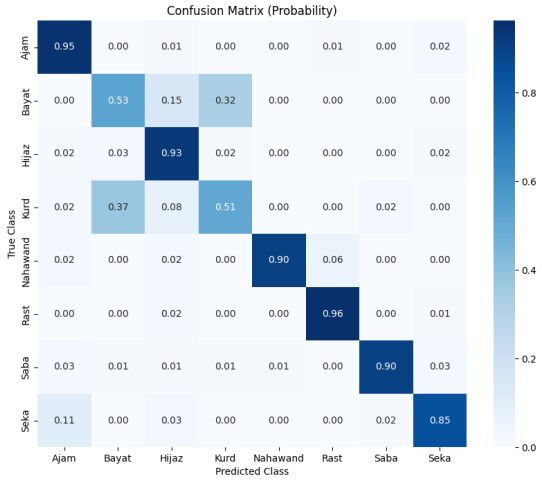

Figure 51: ANN MFCC-means $dataset_2$ confusion Matrix

7.2.2. Chroma mean

We trained the ANN model on the Chroma mean on the two datasets.
For $dataset_1$:
Accuracy = 74.29%
Table 7 shows the model architecture:

Table 7: ANN Chroma-means $dataset_1$ model architecture

Layer	Input	Output	Dropout	BN	AF
1-fc	12	2048	0.2	2048	RELU
2-fc	2048	1024	0.2	1024	RELU
3-fc	1024	512	0.1	512	RELU
4-fc	512	256	0.1	256	RELU
5-fc	256	128	0	128	RELU
6-fc	128	64	0	64	RELU
output	64	8	0	no	softmax

Hyper parameters:
learning rate = 0.0001
number of epochs = 600
batch size = 64
Figure 52 shows training & validation loss.
Figure 53 shows training & validation accuracy.
Figure 54 shows the Confusion matrix.

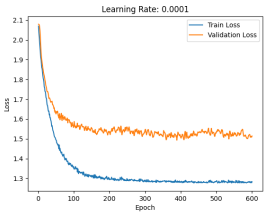

Figure 52: ANN Chroma-means $dataset_1$ loss

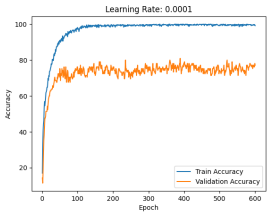

Figure 53: ANN Chroma-means $dataset_1$ accuracy

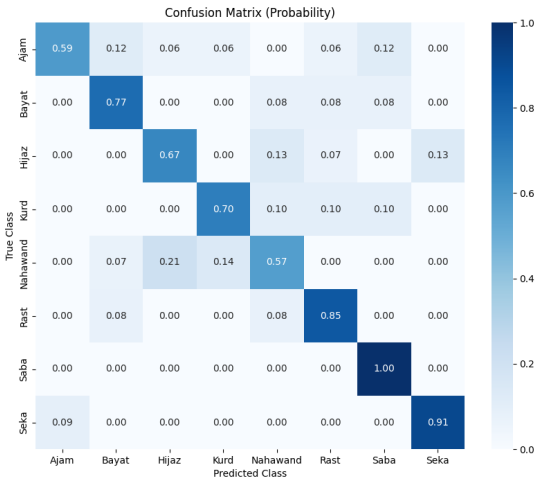

Figure 54: ANN Chroma-means $dataset_1$ confusion Matrix

For $dataset_2$:

Accuracy = 82.09983%
Table 8 shows the model architecture:

Table 8: ANN Chroma-means $dataset_2$ model architecture

Layer	Input	Output	Dropout	BN	AF
1-fc	12	2048	0.2	2048	RELU
2-fc	2048	1024	0.2	1024	RELU
3-fc	1024	512	0.1	512	RELU
4-fc	512	256	0.1	256	RELU
5-fc	256	128	0	128	RELU
6-fc	128	64	0	64	RELU
output	64	8	0	no	softmax

Hyper parameters:
learning rate = 0.0001
number of epochs = 80
batch size = 64
Figure 55 shows training & validation loss.
Figure 56 shows training & validation accuracy.
Figure 57 shows the Confusion matrix.

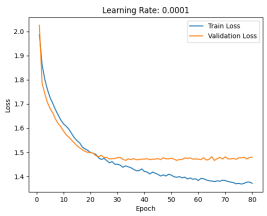

Figure 55: ANN Chroma-means $dataset_2$ loss

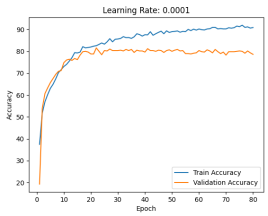

Figure 56: ANN Chroma-means $dataset_2$ accuracy

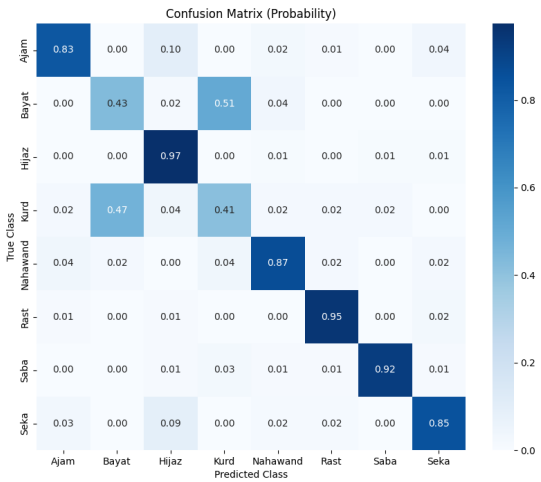

Figure 57: ANN Chroma-means $dataset_2$ confusion Matrix

7.2.3. MFCC mean + Chroma mean

We trained the ANN model on the Chroma mean on the two datasets.

For $dataset_1$:

Accuracy = 92.14%
Table 9 shows the model architecture:

Table 9: ANN MFCC & Chroma means $dataset_1$ model architecture

Layer	Input	Output	Dropout	BN	AF
1-fc	32	2048	0.25	2048	RELU
2-fc	2048	1024	0.25	1024	RELU
3-fc	1024	512	0.1	512	RELU
4-fc	512	256	0.1	256	RELU
5-fc	256	128	0	128	RELU
6-fc	128	64	0	64	RELU
output	64	8	0	no	softmax

Hyper parameters:
learning rate = 0.0001
number of epochs = 250
batch size = 64
Figure 58 shows training & validation loss.
Figure 59 shows training & validation accuracy.
Figure 60 shows the Confusion matrix.

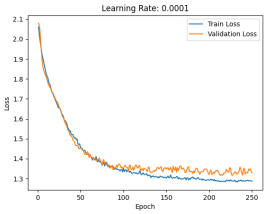

Figure 58: ANN MFCC & Chroma means $dataset_1$ loss

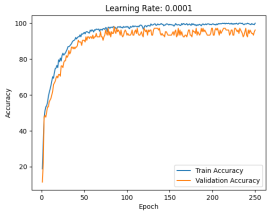

Figure 59: ANN MFCC & Chroma means $dataset_1$ accuracy

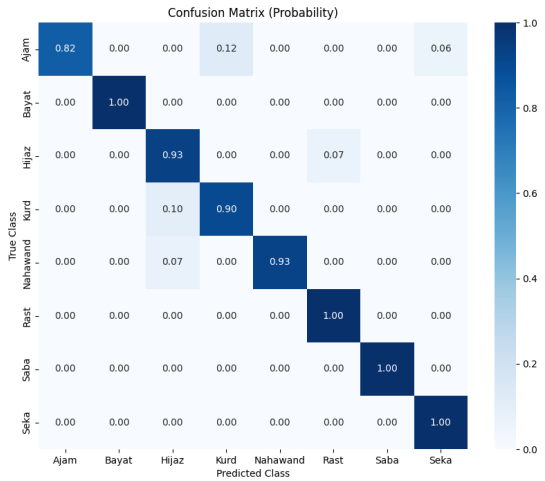

Figure 60: ANN MFCC & Chroma means $dataset_1$ confusion Matrix

For $dataset_2$:

Accuracy = 86.40275%
Table 10 shows the model architecture:

Table 10: ANN MFCC & Chroma means $dataset_2$ model architecture

Layer	Input	Output	Dropout	BN	AF
1-fc	32	4096	0.25	4096	RELU
2-fc	4096	1024	0.25	1024	RELU
3-fc	1024	512	0.1	512	RELU
4-fc	512	256	0.1	256	RELU
5-fc	256	128	0	128	RELU
6-fc	128	64	0	64	RELU
output	64	8	0	no	softmax

Hyper parameters:
learning rate = 0.0001
number of epochs = 250
batch size = 64
Figure 61 shows training & validation loss.
Figure 62 shows training & validation accuracy.
Figure 63 shows the Confusion matrix.

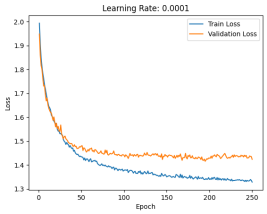

Figure 61: ANN MFCC & Chroma means $dataset_2$ loss

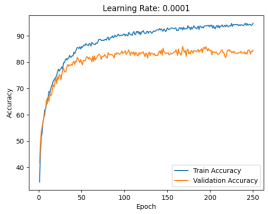

Figure 62: ANN MFCC & Chroma means $dataset_2$ accuracy

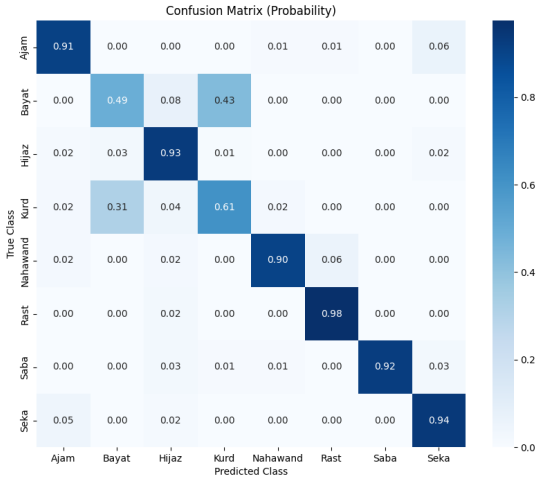

Figure 63: ANN MFCC & Chroma means $dataset_2$ confusion Matrix

7.2.4. MFCC mean + Chorma mean of the mean

For $dataset_1$:

Accuracy = 93.33333%
Table 11 shows the model architecture:

Table 11: ANN MFCC-means & Chroma-mean of the mean $dataset_1$ model architecture

Layer	Input	Output	Dropout	BN	AF
1-fc	21	4096	0.25	4096	RELU
2-fc	4096	1024	0.25	1024	RELU
3-fc	1024	512	0.1	512	RELU
4-fc	512	256	0.1	256	RELU
5-fc	256	128	0	128	RELU
6-fc	128	64	0	64	RELU
output	64	8	0	no	softmax

Hyper parameters:
learning rate = 0.0001
number of epochs = 125
batch size = 64
Figure 64 shows training & validation loss.
Figure 65 shows training & validation accuracy.
Figure 66 shows the Confusion matrix.

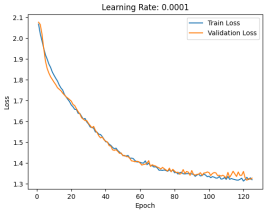

Figure 64: ANN MFCC-means & Chroma-mean of the mean $dataset_1$ loss

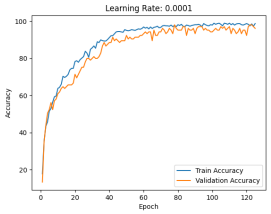

Figure 65: ANN MFCC-means & Chroma-mean of the mean $dataset_1$ accuracy

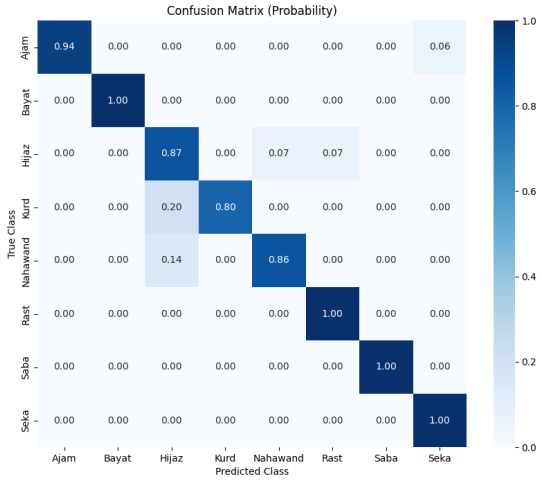

Figure 66: ANN MFCC-means & Chroma-mean of the mean $dataset_1$ confusion Matrix

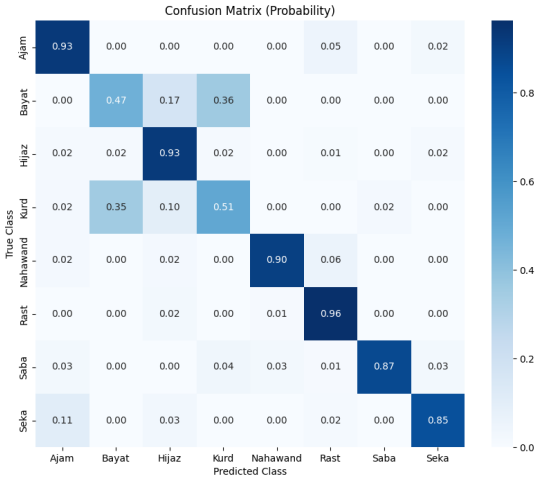

Figure 69: ANN MFCC-means & Chroma-mean of the mean $dataset_2$ confusion Matrix

For $dataset_2$:

Accuracy = 83.47676%
Table 12 shows the model architecture:

Table 12: ANN MFCC-means & Chroma-mean of the mean $dataset_2$ model architecture

Layer	Input	Output	Dropout	BN	AF
1-fc	21	4096	0.25	4096	RELU
2-fc	4096	1024	0.25	1024	RELU
3-fc	1024	512	0.1	512	RELU
4-fc	512	256	0.1	256	RELU
5-fc	256	128	0	128	RELU
6-fc	128	64	0	64	RELU
output	64	8	0	no	softmax

Hyper parameters:
learning rate = 0.0001
number of epochs = 125
batch size = 64
Figure 67 shows training & validation loss.
Figure 68 shows training & validation accuracy.
Figure 69 shows the Confusion matrix.

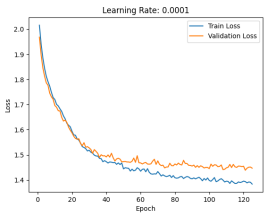

Figure 67: ANN MFCC-means & Chroma-mean of the mean $dataset_2$ loss

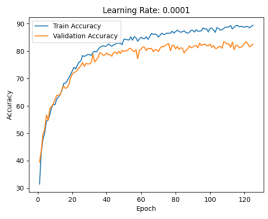

Figure 68: ANN MFCC-means & Chroma-mean of the mean $dataset_2$ accuracy

7.2.5. 7 Features

In the following approach we tried to learn on all the 7 features which are:
MFCC mean, Chorma mean, Spectral bandwidth, Spectral Roll-off, Spectral Centroid, ZRC, and RMS. For $dataset_1$:

Accuracy = 87.61905%
Table 13 shows the model architecture:

Table 13: ANN 7 features $dataset_1$ model architecture

Layer	Input	Output	Dropout	BN	AF
1-fc	37	1024	0.2	1024	RELU
2-fc	1024	512	0.1	512	RELU
3-fc	512	256	0.1	256	RELU
4-fc	256	128	0	128	RELU
5-fc	128	64	0	64	RELU
output	64	8	0	no	softmax

Hyper parameters:
learning rate = 0.0001
number of epochs = 150
batch size = 64
Figure 70 shows training & validation loss.
Figure 71 shows training & validation accuracy.
Figure 72 shows the Confusion matrix.

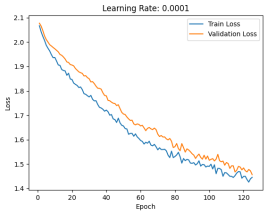

Figure 70: ANN 7-features $dataset_1$ loss

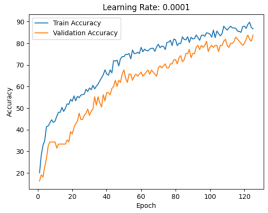

Figure 71: ANN 7-features $dataset_1$ accuracy

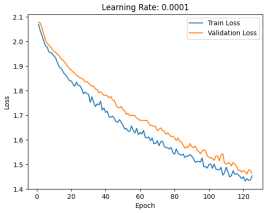

Figure 73: ANN 7-features $dataset_2$ loss

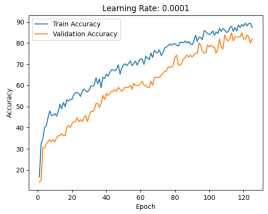

Figure 74: ANN 7-features $dataset_2$ accuracy

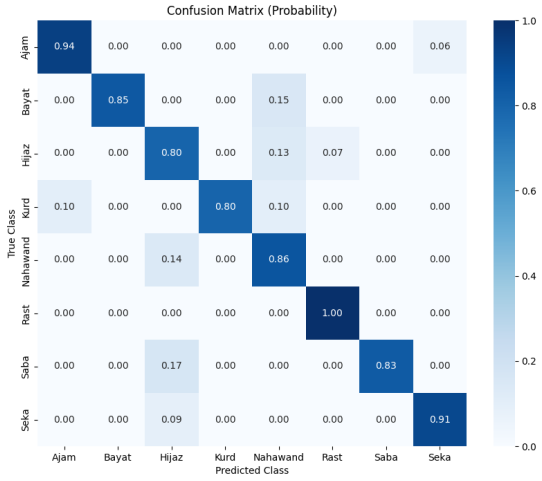

Figure 72: ANN 7-features $dataset_1$ confusion Matrix

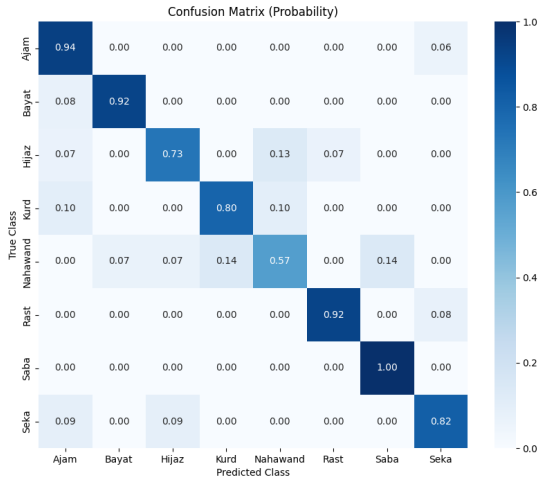

Figure 75: ANN 7-features $dataset_2$ confusion Matrix

For $dataset_2$:

Accuracy = 83.80952%
Table 14 shows the model architecture:

Table 14: ANN 7 features $dataset_2$ model architecture

Layer	Input	Output	Dropout	BN	AF
1-fc	37	4096	0.25	4096	RELU
2-fc	4096	1024	0.25	1024	RELU
3-fc	1024	512	0.1	512	RELU
4-fc	512	256	0.1	256	RELU
5-fc	256	128	0	128	RELU
6-fc	128	64	0	64	RELU
output	64	8	0	no	softmax

Hyper parameters:
learning rate = 0.0001
number of epochs = 125
batch size = 64
Figure 73 shows training & validation loss.
Figure 74 shows training & validation accuracy.
Figure 75 shows the Confusion matrix.

8. Future Improvements

In this paper, we have presented a novel approach for classifying the maqamat of Quranic recitations using deep learning. We have used two datasets of audio samples, extracted various features from them, and trained different models to predict the maqam labels. We have evaluated our models using different metrics, such as accuracy, precision, recall, and F1-score, and compared them with some baseline models. We have achieved promising results, but there is still room for improvement and further research. In this section, we suggest some possible ways to improve our work and achieve better results for the maqam classification task. Some of the future improvements are:

- Data improvement: Collecting more data samples from different sources and reciters, and ensuring that they are properly annotated and preprocessed. More data can help the models learn more features and patterns, and reduce the risk of overfitting or underfitting. Augmenting the data by applying some transformations, such as pitch shifting, time stretching, or adding noise, to increase the diversity and robustness of the data.
- More reciters: Including more reciters in the datasets, especially those who have different styles or accents of

recitation. This can help the models generalize better to different variations and nuances of the maqamat, and increase the applicability and usefulness of the work for a wider audience.

- Various maqam base musical notes: Exploring the use of different musical notes or scales as the basis for the maqamat, such as the Turkish, Persian, or Indian scales. These scales may have different intervals, tunings, or microtones than the Arabic scale, and may capture some aspects of the maqamat that are not well represented by the Arabic scale. Comparing the performance of the models using different scales, and seeing which one is more suitable or accurate for the maqam classification task.

- Random forests and bagging methods to combine models: Applying some ensemble learning techniques, such as random forest or bagging, to combine multiple models and improve their accuracy and stability. These techniques can reduce the variance and bias of individual models, and produce a more reliable and robust prediction. Experimenting with different ways of combining the models, such as voting, averaging, or stacking, and seeing which one gives the best results.

- Other features and models: Trying to extract other features from the audio samples, such as mel-spectrogram, pitch contour, or harmonic-percussive separation, and seeing if they can improve the performance of the models. Trying to use other models or architectures, such as convolutional neural networks (CNNs), recurrent neural networks (RNNs), or transformers, and seeing if they can capture more information and complexity from the audio data.

9. Summary and conclusions

We have proposed a novel approach for classifying the maqamat of Quranic recitations using deep learning. We have used two datasets of audio samples, extracted various features from them, and trained different models to predict the maqam labels. We have evaluated our models using different metrics, such as accuracy, precision, recall, and F1-score, and compared them with some baseline models. We have also suggested some possible ways to improve our work and achieve better results for the maqam classification task.
The best model that we obtained was an artificial neural network (ANN) with an accuracy of 94.3%, using the mean of the mel-frequency cepstral coefficients (MFCC) as the input feature. We have found that adding more features to the MFCC mean feature did not improve the performance, but rather deteriorated it. We have also found that two-dimensional features, such as MFCC or chroma spectrograms, did not perform well with convolutional neural networks (CNNs) or long short-term memory networks (LSTMs), as they were too complex to capture the patterns of the maqamat. Moreover, we have observed that the class distribution of the datasets had an impact on the performance, as the more balanced $dataset_1$ yielded better results than the more unbalanced $dataset_2$, despite having fewer samples and less reciters.
Our work demonstrates the potential and feasibility of using deep learning for maqam classification of Quranic recitations, and provides a useful tool for researchers and practitioners in this field. However, there is still room for improvement and further research, as we have discussed in the previous section. We hope that our work will inspire and motivate more studies and applications in this domain.

References

[1] Muhammad Siddiq al Minshawi. *Muhammad Siddiq al-Minshawi - Mujawwad - Audio - Quran Central*. Quran Central, 2021. URL `https://qurancentral.com/audio/muhammad-siddiq-al-minshawi-mujawwad/`.

[2] Jalal al-Din Al-Suyuti. *Al-Itqan fi Ulum al-Quran*. Dar al-Kutub al-Ilmiyyah, 2007.

[3] A. Y. Ali. *The Meaning of the Holy Qur'an*. Amana Publications, 2004.

[4] Abdul Basit. *Abdul Basit - Mujawwad - Audio - Quran Central*. Quran Central, 2021. URL `https://qurancentral.com/audio/abdul-basit-mujawwad/`.

[5] R. Curtis and S. Dixon. An introduction to digital audio programming using python and wav files. In *Proceedings of the International Conference on New Interfaces for Musical Expression (NIME)*, pages 1–6, 2016.

[6] Stephen Davis and Paul Mermelstein. Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 28(4):357–366, 1980.

[7] T. Fujishima. Realtime chord recognition of musical sound: A system using common lisp music. In *Proceedings of the International Computer Music Conference (ICMC)*, pages 464–467, 1999.

[8] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016.

[9] F. Gouyon and P. Herrera. Exploration of techniques for automatic labeling of audio drum tracks' instruments. In *Proceedings of the MOSART Workshop on Current Research Directions in Computer Music*, 2003.

[10] S. Haykin. *Neural Networks: A Comprehensive Foundation (2nd ed.)*. Prentice Hall, 1999.

[11] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

[12] D. Kriesel. *A Brief Introduction to Neural Networks*. Publisher information not available, 2007.

[13] Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *Nature*, 521(7553): 436–444, 2015.

[14] A. Lerch. *An Introduction to Audio Content Analysis: Applications in Signal Processing and Music Informatics*. Wiley-IEEE Press, 2012.

[15] S. Marcus. *Music in Egypt: Experiencing Music, Expressing Culture*. Oxford University Press, 2007.

[16] Faisal Omari. 699maqam_dataset. Figshare, 7 2023.

[17] Faisal Omari. 3869maqam_dataset. Figshare, 7 2023.

[18] A. V. Oppenheim and R. W. Schafer. *Discrete-Time Signal Processing (3rd ed.)*. Pearson, 2010.

[19] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, Chanan G., and Chintala S. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32 (NeurIPS 2019)*, pages 8024–8035, 2019.

[20] G. Peeters. A large set of audio features for sound description (similarity and classification) in the cuidado project. Technical report, IRCAM, 2004.

[21] quran1. The noble qur'an, 1996.

[22] A. J. Racy. *Making Music in the Arab World: The Culture and Artistry of Tarab*. Cambridge University Press, 2003.

[23] G. Read. *Music Notation: A Manual of Modern Practice*. Taplinger Publishing Company, 1969.

[24] F. Rumsey and T. McCormick. *Sound and Recording: An Introduction (6th ed.)*. Focal Press, 2009.
[25] Sakib Shahriar and Usman Tariq. Maqam478: Qur'anic Recitations in 8 different Maqams. 2 2021. doi: 10.6084/m9.figshare.13489359.v1. URL `https://figshare.com/articles/dataset/Maqam478_Qur_anic_Recitations_in_8_different_Maqams/13489359`.
[26] Sakib Shahriar and Usman Tariq. Classifying maqams of qur'anic recitations using deep learning. *IEEE Access*, 9:117271–117281, 2021. doi: 10.1109/ACCESS.2021.3098415.
[27] Sakib Shahriar and Usman Tariq. Classifying maqams of qur'anic recitations using deep learning. *IEEE Access*, 9:117271–117281, 2021. doi: 10.1109/ACCESS.2021.3098415.
[28] J. O. Smith III. *Mathematics of the Discrete Fourier Transform (DFT): With Audio Applications (2nd ed.)*. W3K Publishing, 2007.
[29] H. H. Touma. *The Music of the Arabs*. Amadeus Press, 1996.
[30] G. Tzanetakis and P. Cook. Musical genre classification of audio signals. *IEEE Transactions on Speech and Audio Processing*, 10(5):293–302, 2002.
[31] G. Van Rossum and F. L. Drake Jr. *Python 3 Reference Manual*. CreateSpace, 2009.
[32] Z. Zhang. A survey on deep learning techniques for image and video semantic segmentation. *Applied Sciences*, 10(5):1854, 2020.

About Us

Project Background

The authors embarked on this project as part of their final year requirements for the Computer Science B.Sc. program. The project involved extensive research, data analysis, programming, and testing to address the challenges posed by maqam classification of Quranic recitations.

Supervisor

The project is conducted under the guidance and supervision of Mr. Nimrod Peleg, a faculty member at the University of Haifa's Computer Science Department and a faculty member at the Department of Electrical Engineering, Technion-Israel Institute of Technology. His expertise and insights have greatly contributed to the success of this project.

Authors

Faisal Omari and Mayas Ghantous are both computer science students pursuing their B.Sc. degrees at the University of Haifa.

Faisal Omari
Receiving his Computer Science B.Sc. (2023) at the age of 19
Student at Etgar Program for excellent students
University of Haifa
GitHub:
LinkedIn:

Mayas Ghantous
Computer Science B.Sc.
Student at Etgar Program for excellent students
University of Haifa
GitHub:
LinkedIn:

Contact Information

For inquiries or further information about this project, you can contact us via email:
Faisal Omari: faisalomari321@gmail.com