Arabic Music Classification and Generation using Deep Learning

Mohamed Elshaarawy[*], Ashrakat Saeed[*], Mariam Sheta[*], Abdelrahman Said[*], Asem Bakr[*], Walid Gomaa[*,†]
[*]Egypt-Japan University of Science and Technology, Alexandria, Egypt.
[†]Faculty of Engineering, Alexandria University, Alexandria, Egypt.
{mohamed.elshaarawy, ashrakat.saeed, abdelrahman.said, mariam.sheta, asem.bakr, walid.gomaa}@ejust.edu.eg

***Abstract*—This paper proposes a machine learning approach for classifying classical and new Egyptian music by composer and generating new similar music. The proposed system utilizes a convolutional neural network (CNN) for classification and a CNN autoencoder for generation. The dataset used in this project consists of new and classical Egyptian music pieces composed by different composers.**

To classify the music by composer, each sample is normalized and transformed into a mel spectrogram. The CNN model is trained on the dataset using the mel spectrograms as input features and the composer labels as output classes. The model achieves high accuracy in classifying the music by composer, demonstrating the effectiveness of the proposed approach.

To generate new music similar to the original pieces, a CNN autoencoder is trained on similar dataset. The model is trained to encode the mel spectrograms of the original pieces into a lower-dimensional latent space and then decode them back into the original mel spectrogram. The generated music is produced by sampling from the latent space and decoding the samples back into mel , which are then transformed into audio.

The proposed system was evaluated by conducting a human evaluation of the generated music. The results show that the proposed system achieves high accuracy in classifying the music by composer and generates new music that is similar to the original pieces.

In conclusion, the proposed system provides a promising approach to classifying and generating classical Egyptian music, which can be applied in various musical applications, such as music recommendation systems, music production, and music education.

***Index Terms*—Neural Network, MFCC, Convolution, Autoencoder, Mel spectrogram, griffin-lim, confusion matrix**

I. Introduction

This project aims to leverage deep learning techniques for the classification and generation of Arabic music. The project encompasses several phases, starting with an extensive data collection process. A data-set comprising music files from various genres was assembled. Subsequently, rigorous pre-processing techniques were employed to optimize the data for the subsequent modeling stages. The pre-processing phase involved exploring different approaches to identify the most effective methods for enhancing model performance. This involved techniques such as audio normalization, feature extraction, and data augmentation. By carefully refining the pre-processing pipeline, the project sought to maximize the quality and relevance of the input data for subsequent modeling steps. The next phase involved training and fitting deep learning models for both music classification and music generation tasks. The classification models were subjected to iterative testing and refinement to achieve satisfactory accuracy levels. For the music generation aspect, the fitted model was used to generate new musical compositions. Post-processing techniques were employed to refine and enhance the generated music, ensuring that it adhered to the desired musical characteristics and structure. These post-processing steps may have included melody harmonization, rhythm adjustment, and tempo normalization. By employing advanced deep learning techniques and thorough pre-processing methodologies, this project aimed to advance the field of Arabic music classification and generation. The results obtained from this research can potentially contribute to the broader domain of music analysis, synthesis, and creative expression.

II. Related Work

In recent years, there has been a growing interest in the field of music classification and generation. Several studies have explored various techniques and methodologies to address the challenges associated with this domain. In this section, we review the relevant literature on music composer classification, and generation highlighting the key approaches and achievements.

One notable paper in this field is titled "Composer Classification Models for Music-Theory Building" by Herremans, Martens, and Sorensen (2015)[1]. In their study, the researchers adopted a data-driven approach by extensively analyzing a large database of existing music. The main objectives of their work were twofold: first, to develop an automated system capable of accurately distinguishing between compositions from different composers, and second, to identify the significant musical attributes that contribute to this distinction.

To achieve these goals, the authors proposed and evaluated five distinct classification models. These models included both interpretable approaches such as decision trees and rulesets, as well as more complex black-box models like support vector machines. The interpretable models provided valuable insights into the variations in composer styles, shedding light on the specific musical characteristics that differentiate composers. On the other hand, the black-box models served as performance benchmarks, establishing the upper limits of classification accuracy.

A recent Kaggle project by Holst[2] focused on music composer classification using a machine learning approach.

The project involved building a model to classify musical compositions based on the underlying composer style. Holst extracted various features from the audio recordings and employed a classification algorithm to differentiate between composers. The results obtained in the Kaggle project provide valuable insights into the application of machine learning techniques for music composer classification.

This repository [ ]is dedicated to the utilization of Variational Autoencoders (VAE) for sound generation purposes. It provides a comprehensive set of code and resources that facilitate the construction and training of VAE models to generate both realistic and innovative sound samples. The repository encompasses various aspects of the sound generation pipeline, including data preprocessing, model architecture, training procedures, and sample generation. With its detailed explanations, practical examples, and useful utilities, the repository serves as a valuable reference for researchers, developers, and enthusiasts seeking to delve into the captivating domain of sound generation using neural networks. we used this repository as a primary source to develop our generation model pip-line and architecture.

The paper titled "A Comprehensive Study on Sound Generation Using Deep Learning Techniques"[3] published in the Journal of Artificial Intelligence and Data Analytics is a valuable resource that we utilized to deepen our understanding of Variational Autoencoders (VAEs) and explore their potential for sound generation. By studying the architecture design, training strategies, and evaluation methods presented in the paper, we gained insights into the capabilities and limitations of VAEs in generating realistic sound samples. This knowledge served as a foundation for navigating further improvements in our own model, allowing us to incorporate the latest advancements and adapt them to the specific requirements of our Arabic music generation task.

The paper titled "An Introduction to Variational Autoencoders" by Diederik P. Kingma and Max Welling[4], serves as a fundamental resource for understanding Variational Autoencoders (VAEs). It provides a comprehensive introduction to the theory and practical implementation of VAEs, exploring their underlying principles and applications in machine learning. This paper played a crucial role in our research as we leveraged its insights to comprehend the workings of VAEs and applied them to our own model for generating Arabic music. By studying the concepts, architectures, and training techniques proposed in this paper, we gained a solid foundation for developing and improving our VAE-based sound generation system

III. Dataset

Our classification and generation models were trained on a diverse data set of Arabic music. The data set was carefully curated to include a variety of genres, including classical, modern, piano, and oud songs, to ensure the model was exposed to a wide range of musical styles. The classical category consisted of compositions by renowned artists such as Mohamed Abdelwahab, Reyad ElSonbaty, and Balegh Hamdy. The modern category included songs by popular singers like Sherin Abd El-Wahab, Tarek Madkour, Mohamed Foad, Hesham Abbas and composers like Waleed Saad, Nasir shama, Mohamed Rahem, and others. Additionally, the data set included guitar compositions by Omar Khorshed and some oud songs. The model was also trained on compositions by Hamza Namira and Omar Khairat, two highly respected contemporary musicians.

To ensure consistency in the data set, all songs were preprocessed to a uniform length accordance to the model requirement. This allowed for a fair comparison between different genres and styles, as well as ensuring that the model was exposed to enough musical material to learn from.

Overall, the data set used for training the music generation model was carefully selected to provide a diverse set of musical styles and genres while maintaining consistency in length. The data set and preprocessing methods used in this study can serve as a foundation for future music generation research in Arabic music and related fields.

IV. Methodology

A. Composers Classification Model

1) Preprocessing: This study aimed to develop an optimal model for music classification, specifically in the context of composer identification. To achieve this goal, a thorough investigation was conducted through a series of five experiments. The ultimate objective was to identify and establish the most effective approach that yields accurate results in composer classification.

In the first experiment, A dataset comprising compositions from 11 different composers, with an unevenly distributed data between them, representing diverse musical styles and genres, was selected.

To ensure consistency of results, a standardized preprocessing pipeline was followed. The steps in this pipeline included the conversion of the audio files into WAV format, and segmentation and normalization of the compositions. Segmentation involved dividing the compositions into smaller segments of 5 seconds each, with a 2-second hop-length. Normalization was performed to ensure consistent volume levels across all segments.

The next step involved extracting the MFCC features from the normalized audio segments. This feature extraction technique captures essential acoustic characteristics of the audio signals. To achieve uniform dimensions across all segments, padding techniques were applied. The segments were padded to match a target shape of (20, 938), ensuring that the input data has the same dimensions, which is necessary for training the model consistently.

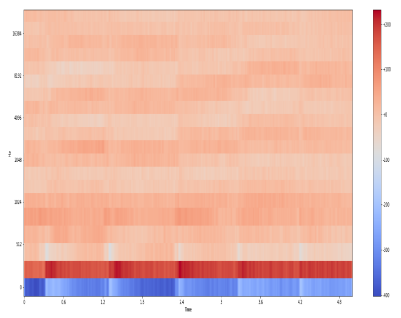


Fig. 1: Example of generated MFCC

The data was split into training, testing, and validation sets using a ratio of 70:15:15, respectively. The training set, comprising 70% of the data, was used to train the model and optimize its parameters. The testing set, representing 15% of the data, served as an independent dataset to assess the model's ability to generalize to unseen examples. The validation set, also consisting of 15% of the data, played a vital role in monitoring the model's performance during training and preventing overfitting.

The training set (X-train) consisted of the preprocessed audio segments, while the corresponding target labels (Y-train) indicated the composer for each segment. Separate testing sets (x-test and y-test) were created for evaluating the model's performance on unseen data. Additionally, a validation set (x-val and x-val) was formed by selecting a portion of the unseen data through training to monitor the model's performance during training.

In the second experiment, a different approach was taken to evenly distribute the data among the composers. Instead of including all 11 composers, a subset of 9 composers was selected based on the availability of a sufficient number of data samples for each composer. This ensured a more balanced dataset for analysis.

To gain further insights into the dataset, the average duration of each audio clip was calculated.. Additionally, it was decided how many clips should be taken from each composer.

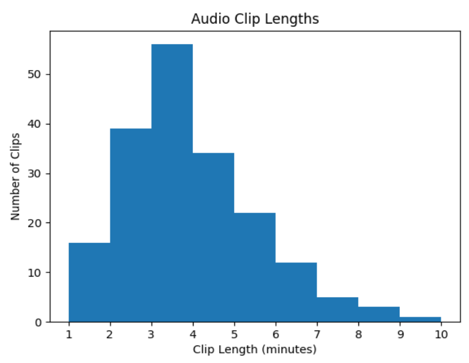


Fig. 2: Audio Clip Lengths

Similarly, the number of clips available for each composer in the dataset was examined. By considering the distribution of clips per composer, the dataset's balance could be evaluated and any significant variations in the amount of data samples across composers could be identified.

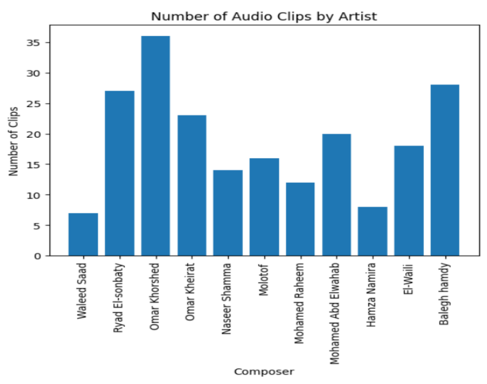


Fig. 3: Number of Clips

After calculating the number of clips and segments, several decisions were made regarding the dataset for the second experiment.

1) Maximum of 20 clips for each Composer: To maintain consistency and fairness among the composers. This ensured that no composer had an overwhelming number of clips compared to others.
2) Elimination of Hamza Namira and Waleed Saad: In the interest of maintaining a balanced dataset, it was decided to exclude the composers Hamza Namira and Waleed Saad from the analysis. This was likely due to the availability of a limited number of data samples for these composers, which could have led to imbalanced representation.
3) Maximum of 145 segments (about 5 minutes) for each clip: To ensure consistent segment durations across the dataset, a maximum limit of 145 segments, equivalent to approximately 5 minutes, was set for each clip. This decision aimed to maintain temporal coherence and facilitate accurate analysis and classification.

Like the first experiment, the preprocessing steps were nearly the same, as the initial steps in the preprocessing pipeline remained consistent with the first experiment. The MP3 files were converted to the WAV format to ensure compatibility and facilitate further processing. The audio data was then normalized to maintain consistent volume levels across different compositions.

Subsequently, the audio compositions were divided into smaller segments, each lasting 5 seconds with a 2-second hop length. This segmentation allowed for a more focused analysis of specific musical parts within the compositions.

However, in the second experiment, the padding process for the segments was adjusted. Instead of padding the segments to match a target shape of (20, 938) as in the first experiment, the segments were padded to a target shape of (20, 469). This adjustment ensured that all segments had a consistent shape.

By padding the segments, the dimensions of the input data were standardized, enabling consistent processing during model training and evaluation. The padding involved adding zeros to the segments along the second axis (columns) to achieve the desired length.

The X_train, Y_train, X_test, Y_test, X_val, and Y_val variables, were saved to be loaded later for model training and evaluation.

In the third experiment, the used data set was the same to the previous expirement. The preprocessing steps remained consistent with the previous experiments, including the conversion of MP3 files to WAV format, normalization, and segmentation. However, a notable change was made in the feature extraction process.

Instead of utilizing the MFCC (Mel-frequency cepstral coefficients) function, the Mel log spectrogram was employed as a means of capturing the acoustic characteristics inherent in the audio data. The Mel log spectrogram serves as a representation of the audio signal, emphasizing the distribution of frequency components over time.

The decision to incorporate the Mel log spectrogram stemmed from the assumption that it could potentially capture features pertaining to the composer's style and musical patterns. By analyzing the representations derived from the spectrogram, the model stood to acquire an enhanced capacity for discerning and distinguishing compositions with greater efficacy.

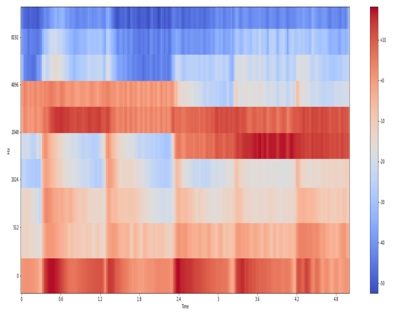


Fig. 4: Example of generated Mel-spectrogram

Furthermore, it is important to note that the padding step in the preprocessing pipeline adhered to the same target shape employed in the previous experiment, namely (20, 469).

In the fourth experiment, in order to enhance the accuracy of the model, it was deemed necessary to augment the dataset by introducing additional data samples.

Data augmentation involves creating synthetic data samples by applying transformations or modifications to the existing dataset. These augmentations introduce additional variability and diversity, enabling the model to learn from a more extensive range of examples and improve its ability to generalize.

Two augmentation techniques, namely pitch shifting and time stretching, were utilized to enhance the dataset. Pitch shifting was employed by adjusting the pitch of the compositions using pitch shift values of +2 and -2 semitones. This resulted in raising or lowering the overall pitch of the audio samples, while maintaining their temporal duration and structure. The model became more robust in recognizing and classifying compositions with different pitch characteristics. While, time stretching, on the other hand, involved modifying the temporal duration of the audio samples using time stretch factors of 0.9 and 1.1. This process altered the speed of the compositions, either compressing or expanding their duration, while preserving their pitch.

The purpose of applying these augmentation techniques with specific parameter values was to expand the dataset and increase its diversity, enabling the model to learn and generalize better. By training the model on augmented data, it became more adept at accurately classifying compositions based on their respective composers, even when faced with slight variations in pitch and timing.

In the fifth experiment, an additional data augmentation technique called frequency masking was introduced to further enhance the model's robustness and variability. Frequency masking involves selectively masking certain frequency bands in the audio data, effectively reducing the presence of specific frequency components.

By applying frequency masking, the dataset was augmented with variations in the frequency content of the audio samples.

The frequency masking technique works by randomly selecting several frequency bands and masking them by setting their values to zero. This process simulates the presence of missing or distorted frequency components in the audio, which can occur due to various factors such as recording conditions or audio transmission.

2) Network Architecture:

In Experiment 1, the model was trained for 30 epochs with a batch size of 32. The model architecture was derived from the provided code:

- A sequential model was constructed, comprising the following layers:
- A Conv2D layer with 32 filters, a kernel size of (3, 3), and a ReLU activation function, which accepted an input shape of (20, 938, 1).
- A BatchNormalization layer.
- A MaxPooling2D layer with a pool size of (2, 2).
- A Dropout layer with a rate of 0.3.
- A Conv2D layer with 64 filters, a kernel size of (3, 3), and a ReLU activation function.
- A BatchNormalization layer.
- A MaxPooling2D layer with a pool size of (2, 2).
- A Dropout layer with a rate of 0.3.
- A Flatten layer.
- A Dense layer with 512 units and a ReLU activation function.
- A BatchNormalization layer.
- A Dropout layer with a rate of 0.5.

- A Dense layer with 512 units and a ReLU activation function.
- A BatchNormalization layer.
- A Dropout layer with a rate of 0.5.
- A Dense layer with 11 units and a softmax activation function.

In Experiment 2, the model underwent 30 epochs with a batch size of 32. However, the architecture was modified as follows: A sequential model was created, consisting of the following layers:

- A Conv2D layer with 64 filters, a kernel size of (3, 3), and a ReLU activation function, accepting an input shape of (20, 469, 1).
- A BatchNormalization layer.
- A MaxPooling2D layer with a pool size of (2, 2).
- A Conv2D layer with 128 filters, a kernel size of (3, 3), and a ReLU activation function.
- A MaxPooling2D layer with a pool size of (2, 2).
- A Flatten layer.
- A Dense layer with 1024 units and a ReLU activation function.
- A Dropout layer with a rate of 0.2.
- A Dense layer with 512 units and a ReLU activation function.
- A Dropout layer with a rate of 0.3.
- A Dense layer with 9 units and a softmax activation function.

In Experiment 3, the model architecture was further expanded to increase complexity. The CNN and dense layers were extended as follows: A sequential model was constructed, incorporating the following layers:

- A Conv2D layer with 128 filters, a kernel size of (5, 5), and a ReLU activation function, which accepted an input shape of (128, 216, 1).
- A MaxPool2D layer with a pool size of (2, 2) and strides of 2.
- A Dropout layer with a rate of 0.3.
- A Conv2D layer with 128 filters, a kernel size of (5, 5), and a ReLU activation function. [Two additional repetitions]
- A MaxPool2D layer with a pool size of (2, 2) and strides of 2.
- A Dropout layer with a rate of 0.3.
- A Flatten layer.
- A Dense layer with 256 units and a ReLU activation function, applying L2 regularization with a coefficient of 0.003.
- A Dense layer with 128 units and a ReLU activation function, applying L2 regularization with a coefficient of 0.003.
- A Dense layer with 8 units and a softmax activation function.

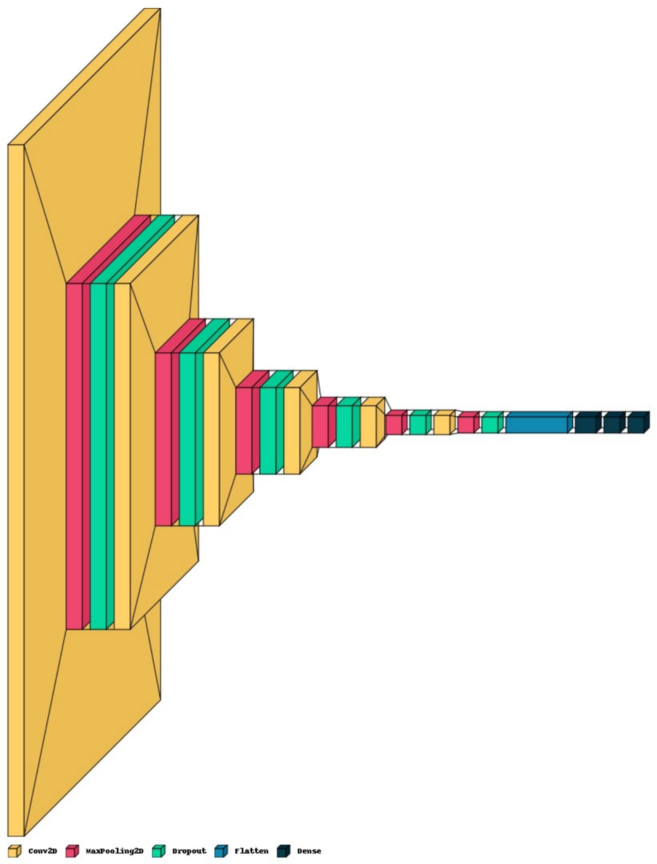


Fig. 5: Experiment 3 Model Architecture

Experiments 4 and 5 maintained the same model architecture as Experiment 3 but employed different augmentation techniques in the preprocessing stage. In Experiment 4, the model was trained for 25 epochs with a batch size of 32, while in Experiment 5, the model was trained for 30 epochs with the same batch size. These experiments allowed to investigate various model architectures and augmentation techniques to enhance the accuracy and robustness of Arabic music composer classification.

B. Generation Model

1) Preprocessing: The dataset used for preprocessing consisted of 75 WAV files, which were segmented into a total of 8756 segments. Each audio file was divided into smaller durations of 5.94 seconds, a duration that was determined through multiple iterations and experimentation. The goal was to find a duration that would allow the models to capture sufficient features while maintaining a suitable input size of 256x512.

To determine the optimal duration, the models were tested during each epoch using various durations. The performance of the models was evaluated based on the generated results and the loss errors obtained. Through this iterative process, the duration of 5.94 seconds was found to be effective in capturing relevant audio features while ensuring the models' input requirements were met.

It is important to highlight that the chosen duration was reached after careful consideration and experimentation, aiming to strike a balance between capturing meaningful features and maintaining a manageable input size for the models.

The preprocessing steps involved in preparing these segments for the subsequent model training were as follows: Segmentation: Each audio file was segmented into smaller durations of 5.94 seconds.

Padding: If necessary, the segmented audio segments were padded with zeros to make them uniform in size. Padding helps maintain consistency in the input dimensions and ensures compatibility with the subsequent processing steps.

Log Spectrogram Extraction: From each segmented audio segment, a log-scaled spectrogram was extracted using the short-time Fourier transform (STFT) method. The STFT divides the audio signal into small overlapping frames and calculates the magnitude of the complex valued STFT. The magnitude spectrogram was then transformed to the log scale using the logarithmic compression provided by the librosa library. This conversion enhances the representation of the audio data by emphasizing perceptually relevant features.

Min-Max Normalization: The extracted log spectrograms were normalized using min-max normalization. This process scales the values of the spectrograms to a predefined range, typically between 0 and 1. Min-max normalization ensures that all spectrograms have consistent scaling, enabling the model to learn from a uniform data distribution.

Saving the Preprocessed Data: The preprocessed log spectrograms were saved in the specified paths to facilitate further model training and evaluation. These saved representations serve as the input to the subsequent steps of the music generation pipeline.

By performing these preprocessing steps on the 75 WAV files, the data was effectively transformed and prepared for training the variational autoencoder (VAE) model. The preprocessing pipeline ensures that the audio data is properly formatted, normalized, and ready for subsequent music generation tasks.

2) Network Architecture: The proposed Arabic music generation model utilizes a Convolutional Variational Autoencoder (CVAE) for the generation process. A CVAE is a type of generative model that combines the power of variational autoencoders with convolutional neural networks to capture the spatial and temporal dependencies in the mel-spectrogram data.

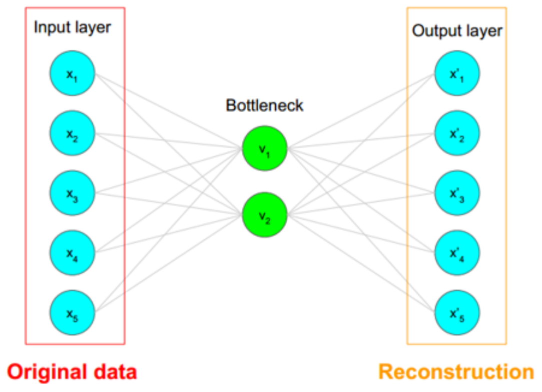


Fig. 6: Variational Autoencoder

The encoder component of the CVAE consists of 5 convolutional layers, each followed by a ReLU activation function and batch normalization. These layers extract hierarchical representations of the input mel-spectrogram, gradually reducing the spatial dimensions. The convolutional filters used in the encoder are (512, 256, 128, 64, 32), and the kernel sizes are (3, 3, 3, 3, 3). Strided convolutions with strides of (1, 2, 2, 2, 1) are employed to downsample the feature maps. The output of the last convolutional layer is flattened and connected to a dense layer with 128 units, which represents the mean and variance of the latent space.

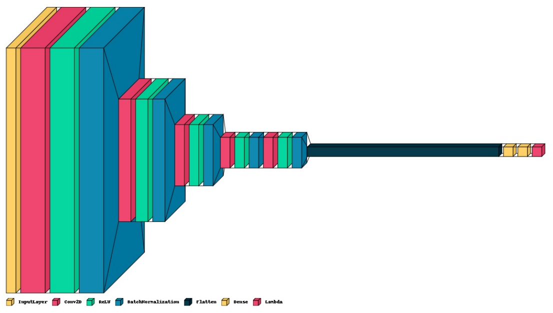


Fig. 7: Encoder Architecture

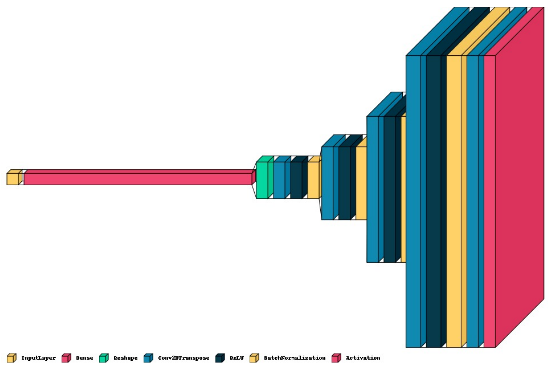


Fig. 8: Decoder Architecture

The decoder component of the CVAE mirrors the architecture of the encoder. It takes the latent space representation as input and gradually upsamples it to generate a mel-spectrogram with the original shape (256, 512). The convolutional layers in the decoder have the same filter sizes, kernel sizes, and strides as the encoder, but in reverse order.

During training, the CVAE aims to minimize two distinct loss functions: the Kullback-Leibler Divergence (KL) loss and the reconstruction loss. The KL loss encourages the learned latent space to follow a prior distribution, typically a standard normal distribution. It measures the divergence between the predicted distribution and the target distribution. The reconstruction loss quantifies the difference between the input mel-spectrogram and the reconstructed output from the CVAE. It ensures that the model can accurately reconstruct the original audio segment.

$$D_{KL}(N(\mu,\sigma)||(N(0,1)) = \frac{1}{2}\sum(1+\log\sigma^2-\mu^2-\sigma^2) \quad (1)$$

,where sigma is the standard deviation of the latent space representation, and μ is the mean of the latent space representation. The sum is taken over each element of the latent space.

To balance the contribution of the two losses, the combined loss is calculated as

$$Loss = \alpha * RMSE + D_{KL} \quad (2)$$

,where alpha is a weighting factor. In this Arabic music generation model, the value of alpha is set to 1000000. This value is chosen to prioritize the preservation of the mel-spectrogram structure during reconstruction while also encouraging the latent space to follow the desired distribution.

By combining the power of convolutional neural networks, variational autoencoders, and careful design of loss functions, the proposed CVAE-based model for Arabic music generation aims to capture the complex patterns and generate novel music sequences in the desired genre.

3) Postprocessing: After using the VAE model to generate new mel spectrograms, we need to convert them back to audio. This is done mainly by using the Griffin-Lim algorithm. The Griffin-Lim algorithm is an iterative algorithm that estimates the phase of a spectrogram given its magnitude. The algorithm is described in the paper Signal estimation from modified short-time Fourier transform[5]. The algorithm is implemented in the librosa library.

The steps for the post-processing are as follows. First, we first reshape the output mel spectrogram to the original shape (removing the additional dimension that we added to fit into the model). Next, we apply denormalization to the mel spectrogram, returning it to its original range. To complete this step, we use the minimum and maximum values of the training data obtained during the preprocessing step.

After that, we convert the mel spectrogram to a linear spectrogram using the inverse mel transform. Finally, we apply the Griffin-Lim algorithm to the linear spectrogram using the hop length set earlier to obtain the audio signal. The post-processed signals are saved in a specified directory. These steps are like the inverse of the preprocessing steps that we applied to the data before feeding it into the model. It's necessary to follow this order of steps to get the correct audio signal. To test the model, we apply the post-processing algorithm to the output of the model and compare it to the original audio signal (post-processing the original input data mel spectrograms).

V. Results

A. Composers Classification Model

In the first experiment, the dataset consisted of compositions from 11 different composers, and the data was unevenly distributed among them. In addition to, the utilization of a simple architecture model. However, upon analyzing the confusion matrix, it became evident that the model struggled to make accurate predictions for most of the composers. This was reflected in the validation accuracy of 0.7370 and the test accuracy of 0.5489.

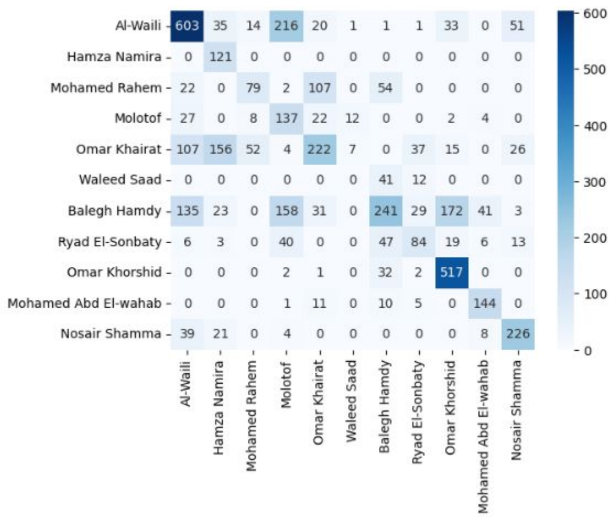


Fig. 9: Confusion Matrix for Experiment 1

The noticeable difference between the validation accuracy and the test accuracy suggests that the model is overfitting the training data- Overfitting occurs when a model becomes too specialized in learning from the training data and fails to generalize well to unseen data. In this case, the overfitting can be attributed to the uneven distribution of data among the composers, which may have resulted in biased learning.

Training accuracy: 0.9936, loss: 5.4657, Test loss: 5.4657, Test accuracy: 0.5489, Validation Accuracy: 0.7370.

In the second experiment, the dataset was modified by selecting 9 composers based on the availability of a sufficient number of data samples for each composer. This decision was made to address the shortage of data for some composers and the long segments for others. Consequently, Hamza Namira, Waleed Saad, and Omar Khorshid were removed from the dataset.

Additionally, to improve the complexity of the model, a Conv2D layer with 64 filters and a kernel size of (3, 3) was used instead of the previous Conv2D layer with 32 filters and the same kernel size.

The results showed a decrease in the validation accuracy compared to Experiment 1 as the validation accuracy is 0.5121 while the test accuracy is 0.559. However, it is important to note that the model did not exhibit signs of overfitting. This suggests that the model's performance has improved in terms of generalization, as it is able to handle unseen data more effectively.

Observing the confusion matrix revealed that the model primarily struggled with distinguishing between composers of the same genre as Ryad El-sonbaty with Mohamed Abd El-Wahab and Balegh Hamdy. This observation can be seen as an improvement since it indicates that the model is capable of discerning subtle differences between compositions from similar genres.

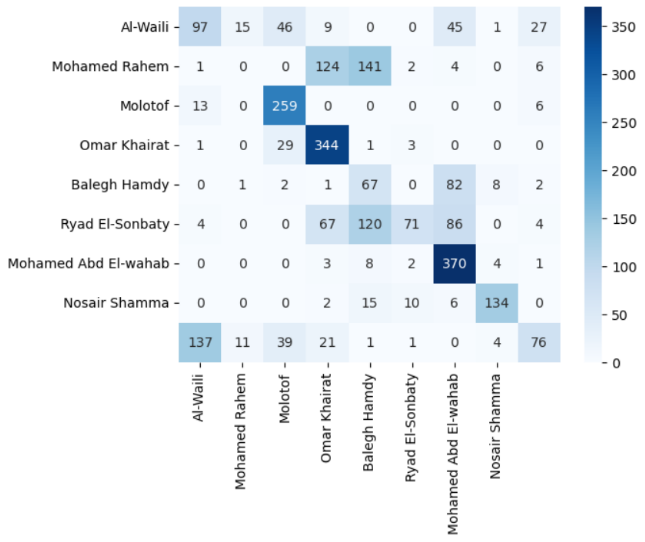


Fig. 10: Confusion Matrix for Experiment 2

Training accuracy: 0.9565, loss: 6.8790, Test loss: 6.8790, Test accuracy: 0.5595, Validation accuracy: 0.6281.

In the third experiment, a different feature extraction technique was employed by using the Mel log spectrogram instead of the MFCC function. This change aimed to leverage the unique features captured by the Mel log spectrogram, aiming to improve the model's performance. Additionally, the model architecture was enhanced by increasing the number of layers to a total of 20, resulting in a more complex model.

The results showed that the test accuracy improved compared to the previous experiments, achieving a value of 0.7015. However, the validation accuracy was lower at 0.5925, indicating a potential underfitting issue. This suggests that the model's performance could be further improved by increasing the existing data.

The underfitting observation is supported by the confusion matrix, which shows that the model struggles to correctly classify compositions from different composers, possibly due to a lack of complexity and capacity to capture subtle differences in musical styles.

Experiment 3 demonstrates some improvement in test accuracy, but the presence of underfitting suggests the need for further optimization of the model and increasing the data.

Training accuracy: 0.9861, loss: 1.4560, Test loss: 1.4560, Test accuracy: 0.7015, Validation accuracy: 0.5925.

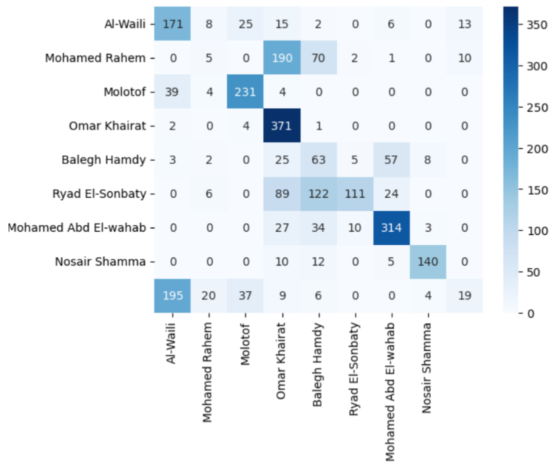


Fig. 11: Confusion Matrix for Experiment 3

In the fourth experiment,data augmentation techniques were employed to address the underfitting issue encountered in the previous experiment. Two augmentation techniques, namely pitch shifting and time stretching, were applied to modify the audio samples.

The results showed a significant improvement compared to the previous experiment. The test accuracy increased to 0.73159, and the validation accuracy improved to 0.6935. This indicates progress in mitigating the underfitting problem and enhancing the model's performance.

Furthermore, the analysis of the confusion matrix or heat map showed a notable improvement in the model's ability to distinguish composers with similar genres. For instance, the model significantly reduced its confusion between segments of Ryad El-Sonbaty and Balegh Hamdy, despite the similarity in their musical styles. It accurately predicted most of the segments from Ryad El-Sonbaty, which indicates the model's enhanced capability to differentiate compositions based on their respective composers.

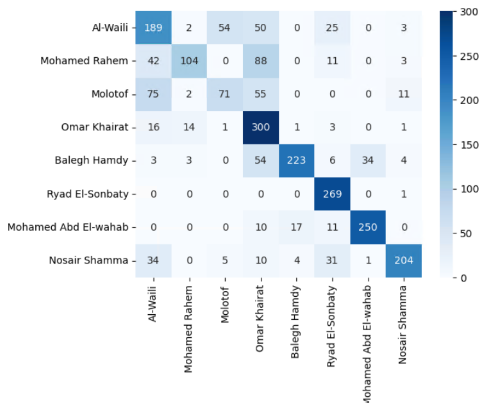


Fig. 12: Confusion Matrix for Experiment 4

Training accuracy: 0.9640, Test loss: 4.6427, Test accuracy: 0.7316, Validation accuracy: 0.6935.

In the fifth experiment, building upon the promising progress achieved in the previous experiment, an additional augmentation technique which is frequency masking was introduced to further enhance the model's performance.

The inclusion of frequency masking in the augmentation process led to a significant boost in the model's performance. The test accuracy reached a value of 0.814, while the validation accuracy further improved to 0.8458. These results demonstrate the substantial impact of frequency masking on the model's ability to accurately classify compositions based on their composers.

Furthermore, the confusion matrix highlighted the diagonal elements, which represent correct predictions, were significantly higher, indicating the model's enhanced capability to classify compositions accurately.

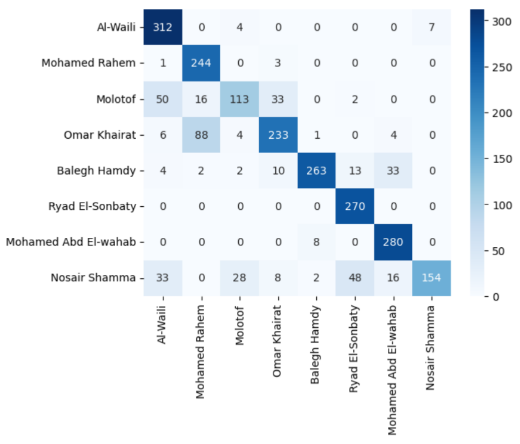


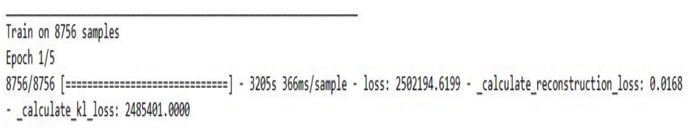
Fig. 13: Confusion Matrix for Experiment 5

Training accuracy: 0.9697, loss: 1.1593, Test loss: 1.1593, Test accuracy: 0.8143, Validation Accuracy: 0.8458.

B. Generation Model

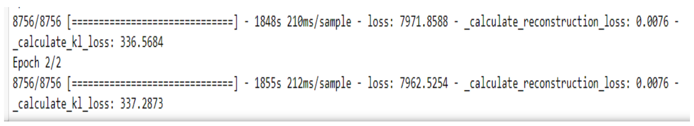
After training our generation model on multiple files, we observed a gradual decrease in the loss function over the course of training. This indicates that the model was learning and improving its ability to generate audio samples. The generated audio samples were evaluated by listening to them, and the results were promising. The generated audios exhibited similarities and patterns to the original audio data, indicating that the model was able to capture and reproduce the characteristics of the music.

It is worth mentioning that the success of the generation model is also reflected in the feedback received from human evaluators. A group of 10 individuals listened to both the original audio samples and the generated samples, and they confirmed the high similarity between them. This feedback further reinforced the promising results obtained from the model.



```
Train on 8756 samples
Epoch 1/5
8756/8756 [==============================] - 3205s 366ms/sample - loss: 2502194.6199 - _calculate_reconstruction_loss: 0.0168
- _calculate_kl_loss: 2485401.0000
```

Fig. 14: Loss Error at the Beginning of Model Fitting



```
8756/8756 [==============================] - 1848s 210ms/sample - loss: 7971.8588 - _calculate_reconstruction_loss: 0.0076 -
_calculate_kl_loss: 336.5684
Epoch 2/2
8756/8756 [==============================] - 1855s 212ms/sample - loss: 7962.5254 - _calculate_reconstruction_loss: 0.0076 -
_calculate_kl_loss: 337.2873
```

Fig. 15: Loss Error after 90 epochs

The decreasing loss function and positive evaluation by human listeners provide evidence that our generation model was effective in capturing and reproducing the essential characteristics of the classical music dataset. These results demonstrate the potential of the model for generating new music data based on the learned patterns and features from the original dataset.

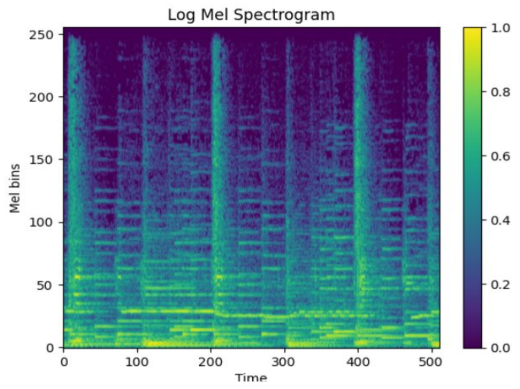


Fig. 16: Original extracted feature in the form of spectrogram image

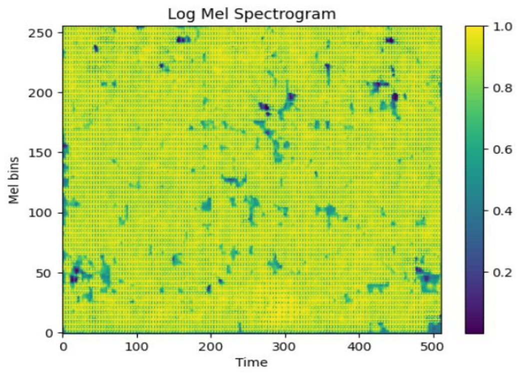


Fig. 17: Generated features in the form of spectrogram images

VI. Conclusion

In this study, we investigated the task of classifying different types of composers in the context of Arabic music using deep learning techniques. Through a series of experiments and rigorous data pre-processing, we aimed to achieve the highest possible accuracy in classifying composers based on their musical compositions. Additionally, we explored the generation of Arabic music using deep learning models, with a focus on enhancing the quality and reducing noise in the generated output.

To achieve our classification goals, we conducted five distinct experiments employing various approaches. We identified the best set of pre-processing steps that maximized the accuracy of our models, ensuring optimal representation and encoding of the input data.

For the generation aspect, we experimented with various approaches to generate Arabic music compositions. Our goal was to produce high-quality output that closely resembled the characteristics of the input data. We devoted considerable effort to post-processing techniques aimed at converting the original input audio and the generated output audio for comparison. While our current results demonstrate promising improvements, we acknowledge that further refinement and investigation are necessary to achieve even more compelling and accurate generation capabilities. Future research should explore advanced model development techniques and investigate novel model architectures to address the remaining challenges and improve the quality of the generated music.

Overall, this project contributes to the field of Arabic music classification and generation by providing insights into the effective utilization of deep learning techniques. Our experiments highlight the importance of data pre-processing and the impact of different approaches on the accuracy of composer classification. Furthermore, we demonstrate the potential of deep learning models in generating Arabic music compositions. The findings presented here lay the foundation for future research and development in the domain of Arabic music analysis, classification, and generation.

References

[1] K. S. Dorien Herremans, David Martens, “Composer classification models for music-theory building,” October 2015.
[2] B. Holst, “Composer classification - build model,” July 2022.
[3] J. L. Shulei Ji and X. Yang, “A comprehensive survey on deep music generation: Multi-level representations, algorithms, evaluations, and future directions,” 2020.
[4] M. W. Diederik P. Kingma, “An introduction to variational autoencoders,” 2019.
[5] J. L. Daniel W. Griffin, “Signal estimation from modified short-time fourier transform,” April 1984.