Automatic Music Generator Using Artificial Intelligence

Thanushree N[1],Yogeshwar P[2], Srileka R[3], Tamilsudar S[4]
[1, 2, 3, 4] Dept of Computer Science and Technology
[1, 2, 3, 4] SNS College of Engineering, Kurumbapalayam(PO),
Coimbatore, Tamil Nadu – 641 107

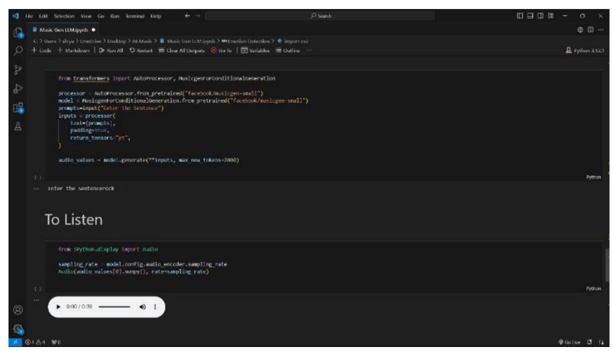
Abstract- *Artificial intelligence (AI)-based automatic music production technology, with a special emphasis on its capacity to produce music that is suited to human moods. Artificial intelligence (AI)-powered automatic music generators are becoming potent instruments for crafting customized musical experiences that speak to the feelings and moods of listeners.*

It delves into the fundamental AI structures and methods used in emotional computing, deep neural networks, and reinforcement learning to create music based on mood. These algorithms use extensive datasets that have been labeled with emotional terms to identify patterns and correlations between musical elements and human emotions. This allows them to create music that elicits particular emotional reactions from listeners. Moreover, the implications and real-world uses of mood-based automatic music generation in a variety of fields. From improving user experiences with games and entertainment to offering therapeutic interventions in.

Keywords- Python, Open CV, LLM module, Media Pipe , Tensor Flow, Keras API, Streamlit

I. INTRODUCTION

The multifaceted applications of Artificial Intelligence (AI) in diverse projects across various domains. AI has emerged as a transformative technology, revolutionizing project management, decision-making processes, and problem-solving approaches. By leveraging advanced algorithms, machine learning techniques, and data analytics, AI enhances efficiency, accuracy, and innovation in project implementation.The specific examples of AI utilization in projects, ranging from predictive analytics and forecasting to natural language processing, computer vision, recommendation systems, autonomous systems, healthcare diagnosis, financial trading, energy management, environmental monitoring, and creative applications. These applications exemplify AI's versatility and effectiveness in addressing complex challenges and optimizing outcomes across different industries and sectors.The role of AI in enabling automation, optimization, and intelligent decision support in project management and execution. AI-powered tools and systems assist project managers and teams in data analysis, risk assessment, resource allocation, and strategic planning, leading to improved project outcomes and organizational performance. Ethical considerations surrounding AI implementation in projects, including concerns related to bias, privacy, transparency, and accountability, are also addressed. The abstract underscores the importance of ethical AI development practices and responsible deployment strategies to mitigate potential risks and ensure equitable and beneficial outcomes for all stakeholders..



II. PYTHON

Python's versatility has made it a go-to choice in various fields, spanning from data science and artificial intelligence to web development and scientific research. In data science, Python's rich ecosystem of libraries, including NumPy, Pandas, and SciPy, provides powerful tools for data manipulation, analysis, and visualization. Its simplicity and readability make it particularly attractive for prototyping and implementing machine learning algorithms using libraries like TensorFlow and scikit-learn.

In web development, Python shines with frameworks like Django and Flask, which streamline the process of building scalable and feature-rich web applications. These frameworks offer robust solutions for backend development, handling tasks such as routing, authentication, and database management efficiently. Moreover, Python's versatility extends to frontend development with libraries like Flask or Django templatesand JavaScript integration.

Python's applications also extend to scientific researchand computational modeling. With libraries like Matplotlib and Seaborn, Python facilitates the creation of publication-qualities plots and charts for visualizing data in scientific publications. Additionally, Python's integration with specialized libraries for symbolic mathematics and computational physics, such as SymPy and PyCuda, enables researchers to tackle complex mathematical problems and simulations effectively.

Python's adoption in education is also notable, thanks to its beginner-friendly syntax and vast community support. Many educational institutions and coding bootcamps use Python as the introductory language for teaching programming concepts, making it accessible to learners of all ages and backgrounds.

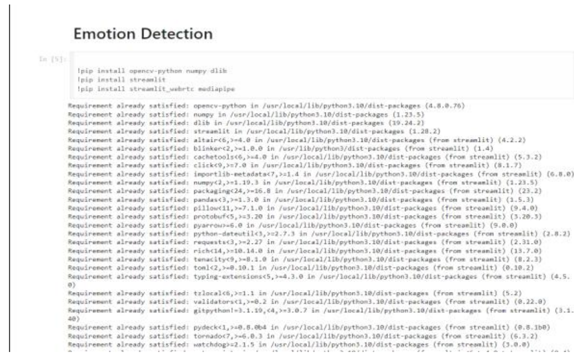
III. LLM MODULE

The LLM (Longformer-Large-Music) module is a variant of the Longformer architecture, a transformer-based model designed to handle long sequences efficiently. In the context of automatic music generation, the LLM module can be used as part of a larger AI system to generate music sequences like data representation, training data, model architecture, music generation process, evaluation and fine-tuning and post processing.

In Data Representation the LLM module requires input data in a tokenized format. In the case of music generation, this could involve representing musical elements such as notes, chords, and rhythms as tokens. Each token represents a specific musical event or feature within the sequence. The Training of Data The LLM module needs to be trained on a large dataset of music sequences to learn the underlying patterns and structures of music. This dataset could consist of MIDI files, audio recordings, or symbolic music representations.

In Music Generation Process .**a**). **Input Encoding** The input music sequence is tokenized and encoded to represent the musical features as input embeddings suitable for consumption by the LLM module. b).**Model Inference**: The encoded input sequence is fed into the LLM module, which processes the sequence through multiple layers of self-attention mechanisms to capture contextual dependencies. **c).Generation** of the trained LLM module is capable of generating music sequences by auto regressively predicting the next token in the sequence based on the previously generated tokens. This process continues iteratively until the desired length of the music sequence is generated.

Evaluation and Fine-Tuning of the generated music sequences can be evaluated based on predefined criteria such as musicality, coherence, and stylistic fidelity. Fine-tuning of the LLM module can be performed based on feedback from human evaluators or objective metrics to improve the quality of generated music.Post-processing of the generated music sequences may undergo post-processing steps to enhance their quality or adapt them to specific musical styles or genres. This could involve smoothing transitions between musical events, applying tempo adjustments, or harmonizing the sequence.



IV. MEDIA PIPE

Media Pipe is an open-source framework for building real-time Multimedia processing pipelines. While Media Pipe itself is not typically used for automatic music generation, it can be integrated into an automatic music generator system to analyze audio or visual input, extract features, or perform other preprocessing tasks. Here's how Media Pipe might be used in conjunction with an automatic music generator:

Audio Analysis:

- Media Pipe's audio processing capabilities can be utilized to analyze Input audio streams, extract features such as pitch, tempo, rhythm, and timbre, and convert them into a format suitable for further processing by the music generator.
- For example, Media Pipe's pre-trained models for audio event detection or pitch estimation could be used to identify musical notes or patterns in the input audio signal.

Gesture Recognition:

- In scenarios where the automatic music generator is designed for interactive or gesture-based control, Media Pipe's hand or pose detection models can be employed to track the movements of the user's hands or body.

Visual Input Processing:

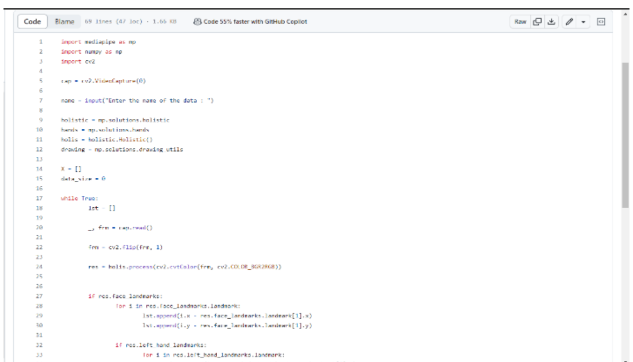
- While not directly related to music generation, Media Pipe'scapabilities in processing visual input, such as images or video frames, could still be relevant in certain contexts.

For example, visual information captured by a camera could be used to trigger or influence the music generation process based on changes in the environment or user interactions captured through MediaPipe's visual processing models.

Synchronization and Integration:

MediaPipe can be integrated into the overall architecture of the automatic music generator as a preprocessing or feature extraction module.

The output from MediaPipe's processing pipelines, whether it's audio features, gesture information, or visual cues, can be fed into the music generation algorithm or system to inform the generation process and produce music that responds dynamically to the input.



```
Code  Blame  69 lines (47 loc) · 1.66 KB   Code 55% faster with GitHub Copilot   Raw

import mediapipe as mp
import numpy as np
import cv2

cap = cv2.VideoCapture(0)

name = input("Enter the name of the data : ")

holistic = mp.solutions.holistic
hands = mp.solutions.hands
holis = holistic.Holistic()
drawing = mp.solutions.drawing_utils

X = []
data_size = 0

while True:
	lst = []

	_, frm = cap.read()

	frm = cv2.flip(frm, 1)

	res = holis.process(cv2.cvtColor(frm, cv2.COLOR_BGR2RGB))


	if res.face_landmarks:
		for i in res.face_landmarks.landmark:
			lst.append(i.x - res.face_landmarks.landmark[1].x)
			lst.append(i.y - res.face_landmarks.landmark[1].y)

		if res.left_hand_landmarks:
			for i in res.left_hand_landmarks.landmark:
```

V. OPENCV

OpenCV (Open Source Computer Vision Library) is primarily designed for computer vision tasks, such as image and video processing, rather than music generation. However, OpenCV can still be utilized in an automatic music generator system in several ways:

Audio Visualization: OpenCV can be used to visualize audio data by converting it into spectrograms, waveforms, or other visual representations. These visualizations can help users understand the characteristics of the audio input, such as frequency content and amplitude variations, which can inform the music generation process.

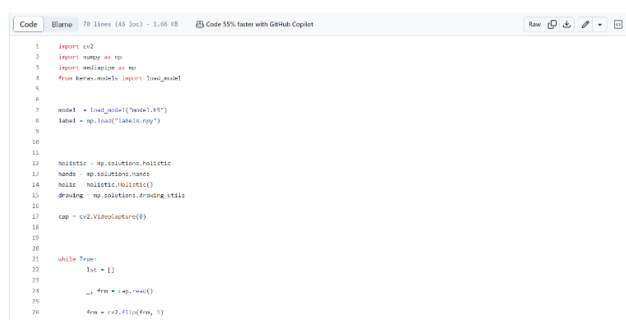
Gesture Detection: OpenCV's object detection and tracking capabilities can be employed to detect gestures or movements from visual input, such as hand movements or body gestures. These detected gestures can then be mapped to specific musical actions or parameters, allowing users to control aspects of the music generation process through visual interactions.

Integration with Other Sensors: OpenCV can be used to process visual input from cameras or other sensors that capture user interactions or environmental cues. For example, if the automatic music generator system is designed for interactive installations or performances, OpenCV can analyze visual input to trigger or influence the music generation process based on changes in the environment or user interactions.

Real-time Processing: OpenCV's efficient and optimized implementation makes it suitable for real-time applications, allowing the automatic music generator to process visual input and generate music on-the-fly with minimal latency. This capability is particularly valuable in interactive music generation scenarios where the system needs to respond quickly to user input or changes in the environment.

Gesture-based Control: By using OpenCV for gesture detection and recognition, users can interact with the music generator system using hand gestures or body movements. For example, users could control parameters such as tempo, volume, or instrument selection by performing specific gestures captured by OpenCV.

User Engagement: Visualizations generated by OpenCV can enhance user engagement with the music generation system by providing real-time feedback on their interactions. This visual feedback can help users understand how their actions are influencing the music being generated, leading to a more immersive and interactive experience.



```
Code  Blame  70 lines (46 loc) · 1.66 KB   Code 55% faster with GitHub Copilot   Raw

import cv2
import numpy as np
import mediapipe as mp
from keras.models import load_model


model  = load_model("model.h5")
label = np.load("labels.npy")



holistic = mp.solutions.holistic
hands = mp.solutions.hands
holis = holistic.Holistic()
drawing = mp.solutions.drawing_utils

cap = cv2.VideoCapture(0)



while True:
	lst = []

	_, frm = cap.read()

	frm = cv2.flip(frm, 1)
```

VI. TENSORFLOW

TensorFlow, as a powerful deep learning framework, can be used in various ways within an automatic music generator system. Here's how TensorFlow can be applied:

Sequence Modeling: TensorFlow can be used to build sequence modeling architectures such as recurrent neural networks (RNNs), long short-term memory networks (LSTMs), or transformers. These models can learn patterns and dependencies within music sequences and generate new sequences based on learned patterns.

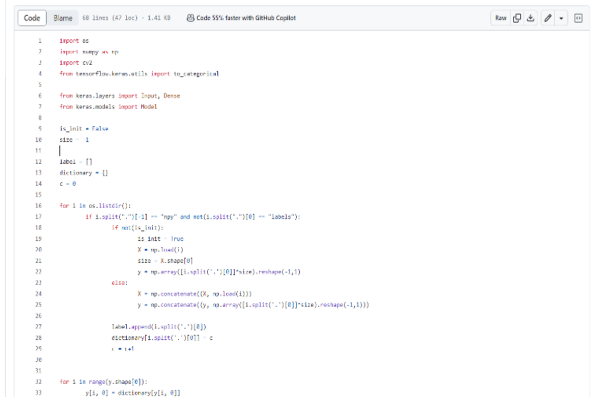
Music Representation Learning: TensorFlow can train deep learning models to learn meaningful representations of music data. For instance, convolutional neural networks (CNNs) can be trained to extract features from raw audio signals or symbolic music representations (e.g., MIDI files), which can then be used as input for music generation models.

Generative Models: TensorFlow can be used to implement generative models such as variational auto encoders (VAEs) or generative adversarial networks (GANs) for music generation. VAEs can learn a latent representation of music data and generate new music samples, while GANs can generate music samples by training a generator network to produce realistic music samples and a discriminator network to distinguish between real and generated samples.

Conditional Generation: TensorFlow enables conditional generation, where the generated music sequences can be conditioned on specific attributes or inputs. For example, the generator network in a conditional GAN can be conditioned on certain musical characteristics (e.g., genre, mood, or style) to generate music samples that match the specified conditions.

Training and Optimization: TensorFlow provides tools for training deep learning models efficiently on large datasets. It includes optimization algorithms, automatic differentiation, and distributed training capabilities, which are crucial for training complex music generation models effectively.

Evaluation Metrics: TensorFlow can be used to implement evaluation metrics for assessing the quality of generated music sequences. For example, evaluation metrics such as perplexity, fidelity, or musicality can be computed using TensorFlow to evaluate the performance of music generation models..




```
import os
import numpy as np
import cv2
from tensorflow.keras.utils import to_categorical

from keras.layers import Input, Dense
from keras.models import Model

is_init = False
size = -1

label = []
dictionary = {}
c = 0

for i in os.listdir():
	if i.split(".")[-1] == "npy" and not(i.split(".")[0] == "labels"):
		if not(is_init):
			is_init = True
			X = np.load(i)
			size = X.shape[0]
			y = np.array([i.split('.')[0]]*size).reshape(-1,1)
		else:
			X = np.concatenate((X, np.load(i)))
			y = np.concatenate((y, np.array([i.split('.')[0]]*size).reshape(-1,1)))

		label.append(i.split('.')[0])
		dictionary[i.split('.')[0]] = c
		c = c+1


for i in range(y.shape[0]):
	y[i, 0] = dictionary[y[i, 0]]
```

VII. KERAS API

Keras is a high-level neural networks API, written in Python and capable of running on top of TensorFlow, among other frameworks. It provides a user-friendly interface for building, training, and deploying deep learning models, including those used in automatic music generation. Here's how Keras API can be used in an automatic music generator:

Model Architecture Definition: Keras allows developers to define the architecture of the neural network models used in music generation. This includes specifying the layers, activation functions, and connectivity patterns of the model. For example, developers can define recurrent neural networks (RNNs) with LSTM layers, convolutional neural networks (CNNs), or more advanced architectures like transformers for sequence modeling tasks.

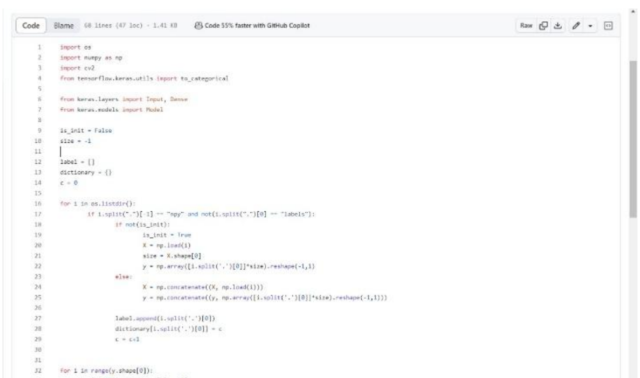
Layer Stacking and Model Composition: Keras enables the stacking of layers to create complex neural network architectures for music generation. Developers can easily add layers such as LSTM, Dense, Conv1D, or Embedding layers to the model and configure their parameters using the Keras API. Additionally, Keras provides functionalities for composing multiple models into a single model, enabling the creation of ensemble models or multi-modal music generators.

Model Compilation: Keras allows developers to compile the defined model by specifying the loss function, optimizer, and evaluation metrics. For music generation tasks, developers can choose appropriate loss functions (e.g., categorical cross-entropy), optimizers (e.g., Adam), and metrics (e.g., accuracy) based on the specific objectives of the music generator.

Model Training: Keras provides high-level APIs for training deep learning models on training data. Developers can use the fit() method to train the model on input music sequences and their corresponding target sequences. Keras handles the training loop internally, including forward and backward propagation, parameter updates, and mini-batch processing, making it easy to train music generation models.

Model Evaluation: After training, developers can evaluate the performance of the trained model using the evaluate() method. Keras computes evaluation metrics such as loss and accuracy on a separate validation dataset to assess the model's generalization performance and identify potential overfitting or underfitting issues.

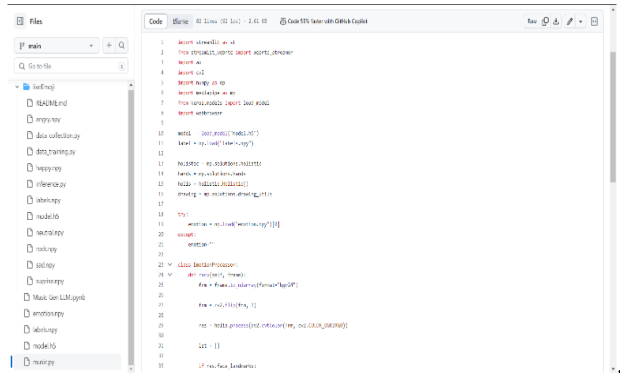
Model Prediction and Generation: Once trained, the Keras model can be used to generate music sequences by predicting the next tokens or events in the sequence iteratively. Developers can use the predict() method to generate new music sequences based on seed sequences or initial conditions. For example, in recurrent neural network models, developers can generate music sequences auto regressively by feeding the model's predictions back as input for subsequent time steps.



VIII. STREAMLIT

Streamlit is an open-source framework that allows developers to create interactive web applications with Python quickly. While Streamlit itself does not generate music, it can be used to create a user interface for interacting with an automatic music generator system. Here's how Streamlit works in the context of an automatic music generator:

Streamlit provides a simple and intuitive way to create web-based interfaces for automatic music generator systems. Developers can use Streamlit to build interactive dashboards or applications that allow users to control and customize the music generation process. For example, developers can create sliders, dropdown menus, or text input fields using Streamlit's widgets to enable users to adjust parameters such as tempo, key, style, or instrumentation.



IX. CONCLUSION

An automatic music generator utilizing AI, Python, OpenCV, Media Pipe, Streamlit, and Keras API offers a comprehensive solution for creating interactive and engaging music generation experiences. Python serves as the backbone, providing a versatile programming environment for integrating various AI tools and frameworks.

OpenCV and MediaPipe enable the system to analyze audio and visual inputs, detect gestures, and capture user interactions. These inputs are then processed and used to inform the music generation process.

Keras API facilitates the development and training of deep learning models for music generation, offering a user-friendly interface for building, training, and deploying neural network architectures.

Streamlit complements the system by providing a web-based interface for users to interact with the music generation process in real-time. Through sliders, dropdown menus, and other widgets, users can control parameters such as tempo, style, and instrumentation, while visualizations and audio playback features enhance the user experience.

Overall, this integrated approach harnesses the power of AI and Python-based technologies to create a seamless and interactive automatic music generator capable of producing diverse and engaging musical compositions.

REFERENCES

[1] Artificial Intelligence Modern Approach by Stuart Russell and Peter Norvig.
[2] Artificial Intelligence 3rd Edition by Patrick Henry Winston

[3] Artificial Intelligence- Structures and Strategies for ‘computing problem solving’ by George F.Luger.