Emotion Based Music Generation

Furkan ÇOLAK

There are 3 python scripts,

1) **Training Script for Emotion Detection (TRAIN_CNN.py)**
2) **Training Script for Music Generation (TRAIN_RNN.py)**
3) **Emotion Detection and Music Generation (GENERATE_MUSIC_BY_FACES.py)**

How to work

1. Run *TRAIN_CNN.py* to generate Trained *facial_emotion_model.h5* file for *GENERATE_MUSIC_BY_FACES.py* They should be same location.
2. Run *TRAIN_RNN.py* to generate Trained *.pkl* and *.h5* file for each emotion to be used at for *GENERATE_MUSIC_BY_FACES.py* They should be same location.
3. Run *GENERATE_MUSIC_BY_FACES.py* to generate music based on your facial expression.

Training Script For Emotion Detection (TRAIN_CNN.py)

This code is a Python script using TensorFlow and Keras to create, train, and save a Convolutional Neural Network (CNN) model for image classification. The specific application appears to be facial emotion recognition, as suggested by the name of the saved model file 'facial_emotion_model.h5'. Here's a breakdown of the code:

Import Libraries: TensorFlow is imported, which is a popular deep learning library. Keras, a high-level neural networks API, is also used here for building and training the model.

Directory Setup: The directory containing the training data is specified. This directory should contain images for training the model.

Image Preprocessing and Augmentation:

- An ImageDataGenerator is created for preprocessing the images. This includes rescaling the pixel values from [0, 255] to [0, 1] for normalization.
- The flow_from_directory method is used to load images from the specified directory (train_dir). Images are resized to 48x48 pixels, processed in batches of 32, and are in grayscale. The class_mode is set to 'categorical', indicating that the task is a multi-class classification.

Building the CNN Model:

- The model is sequential, meaning layers are added in sequence.
- The first layer is a 2D convolutional layer with 32 filters of size 3x3 and ReLU activation function. The input shape is set to (48, 48, 1) which corresponds to the size of the images and 1 channel for grayscale.
- A max pooling layer is added to reduce the spatial dimensions of the output from the previous layer.
- The Flatten layer converts the 2D matrix data to a vector. This is necessary before using fully connected layers.
- A dense (fully connected) layer with 128 units and ReLU activation function is added.
- The final layer is a dense layer with 7 units and a softmax activation function, suitable for multi-class classification (assuming 7 different emotions/classes).

Compile the Model: The model is compiled with the Adam optimizer and categorical crossentropy loss function, which is standard for multi-class classification tasks. The metric for evaluation is accuracy.

Train the Model: The model is trained for 10 epochs using the training data loaded by train_generator.

Save the Model: Finally, the trained model is saved to a file named 'facial_emotion_model.h5'. This file can be loaded later to make predictions without needing to retrain the model.

Overall, the purpose of this code is to train a CNN model to classify images, likely facial expressions, into one of several categories (emotions). The model is trained using a dataset of grayscale images and is saved for future use.

Training Script for Music Generation (TRAIN_RNN.py)

This Python script is designed for music generation using a Long Short-Term Memory (LSTM) neural network, a type of recurrent neural network (RNN) well-suited for sequence prediction problems. The script focuses on processing MIDI files to train a model that can generate music sequences. Here's a breakdown of its components and workflow:

Import Libraries: The script imports necessary libraries including numpy, pretty_midi for handling MIDI files, TensorFlow's Keras for building the LSTM model, and pickle for saving data structures.

Function - get_notes: This function extracts the pitch of each note from the provided MIDI files. It iterates through all the notes in each instrument of each MIDI file and appends their pitches to a list.

Function - prepare_sequences: This function prepares the input and output sequences required for training the LSTM network. It converts the notes into numerical data and creates sequences of a specified length (sequence_length). Each input sequence is mapped to an output note (the next note in the sequence), which is used for training the network. The input is reshaped and normalized, and the output is one-hot encoded.

Function - create_model: This function builds the LSTM model. It consists of three LSTM layers and dropout layers to prevent overfitting. The final layer is a dense layer with a softmax activation function, used for classification among the different possible notes.

Main Process for Training:

- MIDI files are specified. These files are the dataset for training the model.
- The get_notes function is called to extract notes from these MIDI files.
- The number of unique notes (n_vocab) is determined.
- The prepare_sequences function is used to create the input and output sequences for the LSTM network.
- The create_model function is called to build the LSTM model.
- The model is trained using the prepared sequences for 10 epochs with a batch size of 128.

Saving the Model and Note Mappings:

- After training, the model is saved as 'disgust.h5'. This file can be used later to generate music without retraining.
- The note_to_int mapping (which maps notes to integers) is saved using pickle. This is important for decoding the output of the network back into notes when generating music.

The script is set up to train a model specifically for a certain type of emotion (in this case, 'disgust', as indicated by the names of the MIDI files and the saved model). By changing the MIDI files, you can train models for different types of music or emotions. The trained model can then be used to generate new music sequences based on the patterns it has learned from the training data.

Emotion Detection and Music Generation (GENERATE_MUSIC_BY_FACES.py)

This Python script is a integration of facial emotion recognition and music generation. It uses a pre-trained facial emotion recognition model to detect emotions from a webcam feed and then generates music based on the detected emotion using another set of pre-trained models. Here's a detailed breakdown of its components and workflow:

Import Libraries: The script imports necessary libraries including os, cv2 for computer vision tasks, numpy, keras for deep learning models, pretty_midi for handling MIDI files, and pickle for loading saved data structures.

Load Pre-Trained Facial Emotion Recognition Model: The script loads a pre-trained model (facial_emotion_model.h5) designed to recognize facial emotions.

Initialize Haar Cascade for Face Detection: It uses OpenCV's Haar Cascade classifier to detect faces in the webcam feed.

Webcam Capture: The script starts capturing video from the webcam.

Function - generate_notes: This function generates a sequence of music notes using a trained LSTM model. It creates a random start sequence and then iteratively predicts the next note, updating the sequence each time. The function uses diversity to introduce randomness in the prediction process.

Function - create_midi: This function converts the generated sequence of notes into a MIDI file. It creates PrettyMIDI note objects and writes them to a MIDI file.

Main Loop for Processing Webcam Frames:

- The script continuously captures frames from the webcam.
- Each frame is converted to grayscale and passed through the face detection model.
- Detected faces are highlighted, and the region of interest (face area) is extracted and resized.
- The facial emotion recognition model predicts the emotion of the detected face.
- The predicted emotion is displayed on the webcam feed.

Music Generation Triggered by Key Press:

- When the 'g' key is pressed, the script loads the music generation model corresponding to the detected emotion (e.g., happy.h5 for happiness).
- It also loads the note-to-integer mapping (note_to_int) for the specific emotion.
- The generate_notes function is called to generate a sequence of notes, which is then converted into a MIDI file using create_midi.

Exit Condition: The script exits the main loop and closes the webcam feed when the 'q' key is pressed.

Cleanup: The webcam is released, and all OpenCV windows are closed.

Overall, this script is an application of AI in creative domains, combining computer vision and deep learning to create an interactive experience where the user's facial expressions can influence the generation of music.

REFERENCE

Emotion Image Dataset

https://www.kaggle.com/datasets/jonathanoheix/face-expression-recognition-dataset/

Music Dataset

https://bitmidi.com/