A Pipeline for Generation of New Melodies Based on Subject's Facial Emotion Expression Recognition

Claudiu George Andrei
180332423
MSc Artificial Intelligence
Project Supervisor: Prof. Marcus Thomas Pearce
Queen Mary University of London

***Abstract*— It is widely known that facial expressions are the most intuitive indicator of a person's emotional status. The rapid development of technology in recent years has also had an important impact in the way media is consumed, especially digital music, which is known to induce an emotional response in the listener. Various studies indicate a correlation between a subject's emotion and music, suggesting that the latter can have an impact on a person's mood. The framework proposed in this document aims at generating music which corresponds to the emotion predicted by the model. The architectural style for this system is comprised of two microservices that work together: the first model is able to receive the image of a subject's face as input and determine their emotion, the second is responsible for generating new and original music associated to that emotion. This work demonstrates that it is possible to accurately predict (>90%) emotions and merge computational creativity techniques with deep learning to create a system that has a variety of applications, such as in a mental health context, where it could be employed in mental diseases diagnosis and human social/physiological interaction detection. The results of the framework evaluated using a MOS (mean opinion score) measure and generated artifacts are compared to existing music which is known to incite a particular emotion.**

***Index Terms*—emotion detection, music generation, deep learning, MIDI.**

I. Introduction

Facial expressions are a tool humans have always used to manifest their emotional state to others and they are an indication of one's feelings and intentions. In recent years [1] studies have been conducted to explore the possibilities of automatic facial expression detection and its implementation in mental health issues treatment, sociable robotics [2] and other human-computer interaction systems. In computer vision and machine learning, systems have been explored to extract facial features and expressions to build models capable of recognizing emotion: these have evolved from working with lab controlled, small sized datasets using handcrafted techniques or shallow learning, to large scale and ITW (in the wild) datasets, where emotions can be recognized on subjects contained in a bigger scene without being in the center of the frame, thanks to deep learning techniques developed in the last decade. It is important however to gain an understanding of personality and psychology in order to introduce the pipeline presented in this project. Some current systems function with the aim of recognizing 6 basic emotions which have been known for the better part of half a century; Ekman and Friesen [3] defined these to be surprise, happiness, fear, anger, disgust and sadness. Companies around the globe have experienced issues with these definitions as recent, more advanced studies conducted in the fields of neuroscience [4], have introduced the concept of culture-specific emotions which therefore do not apply universally, establishing that better systems with the ability to represent emotion descriptively encompassing complexity. The Facial Action Coding System (FACS) [5], uses a continuous model that represents emotions on a wider spectrum but the categorical models that describe discrete basic emotions are still the most widely used because of the extensive existing studies and intuitive definition of the subject's emotional state, therefore this paper limits the discussion to categorical models. The inspiration for this project comes from the Giorgio Cam A.I. [6] Google experiment, a machine learning program which works with images, recognizes their content by using image recognition to label them and creates short RAP song using the labels for lyrics. This differs from proposed system since the latter aims at generating a melody in midi format, taking this project in the computational creativity space while also including aspects of deep learning and computer vision, allowing for implementation of a broader range of artificial intelligence techniques. The aim of music and musical artists is to incite an emotional response in the listener; hence emotion is an important component for composers. [7] (Rigg, 1940) Demonstrates that a faster tempo has the tendency to inspire happiness as opposed to a slow tempo which provokes the opposite effect; however, modern day automatic music generation systems do not usually take emotion into account, originating in a lack of AI systems which cannot generate music based on a specific emotion. This paper tries to explore the correlation between music elements and emotions: in music psychology it is well established [8] that relations between valence, structural factors and mode are accounted for by compositors to carefully incite a desired emotion. The use of deep learning techniques for this project is motivated by the lackluster performance of handcrafted models, such as rule-based music generation or grammar-based models. A machine learning approach can learn from an arbitrary corpus of music therefore allowing for a single system to be used for a variety of

applications and musical genres. In the pipeline developed this is an important feature as the application is too complex to be described by a brute force manual design and rules learned by the system are more easily generalized in new contexts where inputs may change, as is the case here since the artifact is influenced by the output of the facial expression recognition. Higher level representations are possible with deep learning given the predisposition to process raw unstructured data; both emotion detection and music generation tasks take advantage of this. Similar systems, encompassing both emotion detection and emotional music generation are not very common therefore experimentation and trial and error are needed to obtain acceptable results; control and adaptation of datasets although not optimal are also needed as datasets for this project are scarcely available.

II. Related Work & Background

Systems which mimic the pipeline proposed in this project are currently rare, however this work is related to the achievement of the following main objectives.

- Exploring methods for emotion detection of faces in a controlled dataset through the use of categorical classification.
- Exploring ways of generating music based on emotion labels through the use of RNNs.

A. Deep learning & emotion recognition

Convolutional Neural Networks have been used in many computer vision applications, including face expression recognition: this method became well established at the beginning of the 21st century and it robust to changes in face location and scale. [9] Has introduced the CNN model in the recognition of facial expression addressing the problems of translation, scale invariance and rotation. CNNs are comprised of 3 types of layers: convolutional layers, pooling layers, and fully connected layers. The convolutional layer contains learnable filters: these perform a linear operation which multiplies a set of weights with the input. This layer provides some benefits to the system as it reduces the number of parameters the that need to be learned because of weight sharing in the same feature map, local connectivity which allows to learn correlation amongst nearby pixels and finally shift invariance to the position of the subject. The pooling layer is what follows the aforementioned step, and its purpose is to reduce the feature size and hence computational cost. Some down sampling for translation invariance strategies include average pooling and max pooling: average pooling is a method which smooths out the image making sharp features harder to identify while max pooling is more useful when the background of the image is dark and the points of interest are lighter pixels in the frame. The fully connected layer is the final component of a CNN and is what ensures that all neurons are connected to activations of the previous layers, enabling 2D feature maps to be converted into a 1-dimensional matrix. Several architectures derived from this exist, such as region based CNNs (R-CNN) [10][11] which first extract many region proposals from the input image and label their classes and bounding boxes after which a CNN is used to forward propagate on individual regions to extract features.

B. Music Generation with Deep learning

Traditional neural networks are incapable of remembering the information they are presented with over a period of time: this is the case for any type of sequential information they are fed with and not only limited to music. In sequential data the next piece of information depends on what happens prior. To work around this limitation Recurrent Neural Networks can be used with sequences of information: they differ from CNNs by having an output that feeds data back into the network. The drawback of RNNs is the presence of an undesired property which makes older inputs become lost as newer sequences are fed into the network, this is known as the 'vanishing gradient problem', [12] the older inputs have less influence onto the output as they were encountered too long into the past, therefore, LSTM (long-short-term-memory) is introduced to overcome the issue. The theory behind LSTMs is to predict what information is relevant to the network and hence remember, and what can be ignored. LSTMs are comprised of 2 channels: the top channel allows for information about the state to flow between iterations while the bottom channel uses previous output to help determine what changes to apply to the top channel by using gates. An LSTM cell usually contains three gates referred to as 'forget gate' and 'remember gate' while the last is used to decide the next output for the network. The concept of LSTM for the generation of music appears to have first been used in 2002 [13] thereafter more complex and refined models have been developed such as [14] which combines RNN with RBMs (Restricted Boltzmann Machines) to generate polyphonic music (multiple instruments/notes playing at the same time). Positional constraints have been used to generate musical artefacts also, as demonstrated by Hadjeres & Nielsen in 2017 [15] however with the development of deep learning there has also been an increase in attempts at generating emotional music using deep neural networks. [16] Was one of the earliest to propose an mLSTM (multiplicative LSTM) model with the ability to compose music based on a specific emotion but the dataset used only contains tracks extracted from video games. The methods aforementioned are label based therefore making it impossible to find the datasets used, which have mostly been labelled manually.

C. Data Management and Data Manipulation

Finding the right dataset for this project proved to be a challenge, especially for the music generation part of the pipeline. Papers on emotion detection refer to the dataset used however, often because of the many years passed since their

publication obtaining the datasets used is not always feasible. Furthermore, having an acceptable amount of labeled training data that includes many possible variations is an important aspect of a dataset which needs to be considered in order for the model to perform well in different scenarios. Some of the datasets considered include the [17] TFD (Toronto Face Database) which a collection of various datasets merged into one. The downside of TFD is the number of unlabeled images, containing 112,234 in total but only having 4178 which are labeled, which is not optimal. The dataset is normalized therefore subjects' eyes are the same distance apart and have equivalent coordinates however it was not chosen because the normalization makes it more lab-controlled thus providing less flexibility when performing emotion detection. Other datasets were analyzed such as the SFEW (static facial expressions in the wild) [19] which was created using static frames from a previous dataset comprised of short movie clips: AFEW (acted facial expression in the wild). The SFEW database while containing many different cases of images is limited to a training sample of 958 images; this is not optimal and furthermore, the labels of the testing samples are not released to the public, making this dataset not an option for use in this project. One acceptable dataset is [18] FER2013 database: introduced at the 2013 International Conference on Machine Learning; this dataset was put together by the Google Image search API and contains 28,709 images which have been curated so to adjust the crop region and resize them into 48x48 pixel pictures. FER2013 dataset is praised for having a good balance between lab-controlled and in the wild images. These are divided into 7 categories: 0= angry, 1= disgust, 2= fear, 3= happy, 4= sad, 5 = surprise, 6= neutral. The labels are distributed as shown in Fig 1.

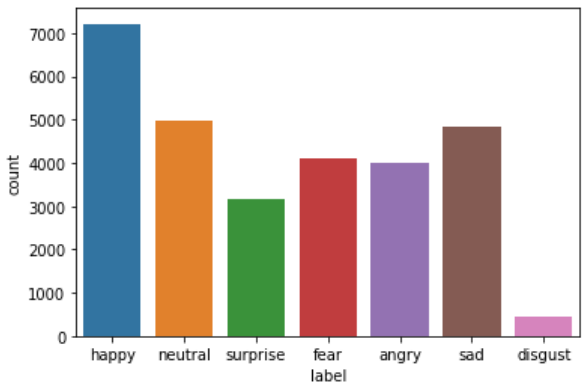

Figure 1 - Label Distribution

Music generation requires data to be presented in a specific format so that features can be extracted from audio excerpts and used to train the model accordingly. Many studies released on the topic of emotional music generation describe their dataset as a collection of manually labelled files which are not made available to the public, making it extremely challenging to find a suitable dataset for this project. The tracks contained in the dataset need to meet certain criteria in order to be considered adequate: they need to be in MIDI format and need to explicitly provoke a certain emotion in the listener. One of the datasets considered is the [21] Lakh Midi v0.1, a collection of 176,581 MIDI files aligned to entries in the famous dataset [20] Million Song Dataset. The files in Lakh MIDI v0.1 are divided by genre and are therefore and because of the extremely big amount of data available it is not feasible to rearrange these by emotion as required for this project. [16] Proposed a method for generating emotion-based tracks, however the dataset for this is not made available and would be irrelevant to this project as the dataset contained music extracted from video games. [22] Is a MIREX like MOOD dataset for emotion classification: it contains 903 audio tracks, 764 lyrics and 196 MIDI files. The intended use for this dataset was that of emotion recognition in music but the MIDI files included can be classified based on emotion. These are contained in directories labelled cluster 0 to 5 with no instruction on which cluster represents which emotion, therefore some curation of the data was needed, and the result obtained is comprised of MIDI files subdivided in 5 folders: angry, happy, neutral, sad and surprised.

III. Preliminaries

A. Pre-Processing

Computer vision datasets may contain features and variations that are not relevant to facial emotions, such as illumination, background and head pose, requiring algorithms to be used in order to normalize the dataset and help the network's performance. Many publicly available datasets for face emotion detection lack the sufficient number of training samples therefore a technique commonly employed is Data augmentation, which ensures that the sufficient training data is available to the network; two forms of data augmentation exist: on the fly and offline. On the fly involves randomly cropping the corners and center of the image while also flipping it horizontally: this generates datasets that can be up to 10 times larger than the original one. The most common data augmentation techniques are offline methods: these apply transformation operation to the images and often include scaling, skewing, rotation, and contrast alterations. These operations generate unseen training samples increasing the overall robustness of the network.

Normalization is another important step for the model to run correctly, a technique applied in the preprocessing part. It aims at changing the values of numeric columns to a more common scale, without changing the range values or losing information. There are many ways of normalizing a dataset however the most commonly used version is the MinMax normalizer, which linearly scales features on a [0,1] interval making the convergence rate much faster. In the emotion detection model

this means that the pixel values of images, which can range from 0 to 255, are transformed using the formula below.

$$z = \frac{x - \min(x)}{[\max(x) - \min(x)]}$$

For the output however using a range of 0 to 1 is not always a good idea because of the activation function that is being used. [23] Explains why a -1 to 1 normalization is preferred in certain tasks. The activation function ReLU uses max(0,x) thus providing better results when negative values are provided also. Another activation function is tanh, which ranges values between -1 and 1. The choice left is sigmoid; however, this does not perform as well as ReLU and tanh: this happens because of the vanishing gradient problem and its nature of being not 0 centered resulting in zig zagged gradient updates for weights.

Why MIDI?

The pipeline proposed for the generation of sentiment driven music uses MIDI (Musical Instrument Digital Interface) files, a symbolic representation for music and a technical protocol for communication between electronic musical instruments. MIDI works by having time stamped messages that convey performance and control data: the main MIDI instructions are Note-on, which indicates that a note is currently playing and also gives information of the channel (track or instrument), note pitch and velocity, and Note-off which indicates that a note should stop playing immediately. This is the industry standard for digital electronic musical instruments and in the domain of Music Representation methods it places in between the acoustic domain and the visual domain. MIDI representation can be rendered into sheet music representation or transformed into audio representations through synthesis performance, the latter is what is used in this project to convert the generated numerical format files into playable audio sequences. MIDI files contain therefore a list of commands to be executed by a synthesizer, making the size much smaller when compared to other audio format: a typical MIDI file is about 40 kilobytes compare to multiple megabytes for WAV files. MIDI also contains information about notes and when they are being played while a WAV file would only contain information about the waveform of a file which contains information about a file's amplitude over a given period of time, making it more difficult to extract information from them.

B. Audio Representation

MIDI files are do not work in Python natively therefore they need to undergo preprocessing steps before they are in a format that is compatible with the algorithms used in the pipeline. In order to convert the files to a format acceptable as input to train the model the [24] MUSIC21 library is used which provides a parser which turns MIDI data into numerical format. Parsed MIDI files are merged into MUSIC21 note objects where the note's pitch and information about ON and OFF events are stored. When parsing a file, the notes are appended in a list while when a chord is encountered (multiple notes playing at once) a dot (.) is used to concatenate pitch values.

C. CNNs with Depthwise Separable Convolutions

In convolutional neural networks the most commonly used convolutional layer are 2D therefore convolutions are performed over multiple input channels at the same time. A new model for CNNs [26] describes how Depthwise convolution keeps each channel separate: it performs a form of factorized convolution by factorizing a regular convolution into a depthwise convolution and a 1 x 1 convolution defined as pointwise convolution. The main advantages of depthwise separable convolution are reduced model size and computation time. While regular convolution filters and combines inputs into outputs in a single step, the depthwise separable convolution divides this into two different layers: one dedicated to filtering and one for combining. This type of convolution was first introduced in [25] and is derived from the GoogleLeNet network style also known as Inception architecture [27] which was developed and optimized based on the Hebbian principle: "*when our brains learn something new, neurons are activated and connected with other neurons, forming a neural network*". As explained in [26] standard convolutional layers take as input a feature map **F** ($D_F \times D_F \times$ M) where D_F represents the width and height of an input feature map and M is the number of input channels. The product is a feature map **G** ($D_G \times D_G \times N$) where D_G is the width and height of an output feature map and N is the channel depth. A standard convolutional layer comprises a kernel **K** ($D_K \times D_K \times M \times N$) where D_K represents the dimensions of the kernel, M is the input channel depth and N is the output depth (number of channels). The output of a standard convolutional layer can be defined as: D

$$\mathbf{G}_{k,l,n} = \sum_{i,j,m} \cdot \mathbf{F}_{k+i-1,\, l+j-1,\, m}$$

Therefore, the computational cost of standard convolutions can be calculated:

$$D_K \cdot D_K \cdot M \cdot N \cdot D_F \cdot D_F$$

Depthwise separable convolution is the combination of depthwise convolution and a 1×1 pointwise convolution where the pointwise is required since depthwise convolution does not combine channels into new features. When combining these two together the cost of Depthwise separable convolution results to be:

$$D_K \cdot D_K \cdot M \cdot D_F \cdot D_F + M \cdot N \cdot D_F \cdot D_F$$

It can be discerned that a reduction in computation is obtained:

$$\frac{D_K \cdot D_K \cdot M \cdot D_F \cdot D_F + M \cdot N \cdot D_F \cdot D_F}{D_K \cdot D_K \cdot M \cdot N \cdot D_F \cdot D_F}$$

$$= \mathbf{1/N + 1/D^2_K}$$

This uses 8 to 9 times less computation than standard methods and is the reason why this style of efficient CNN is used for the emotion recognition model.

IV. System Description

A. Emotion Detection model:

The pipeline designed for this project is comprised of two different networks: one responsible for the recognition of emotions from faces contained in the Google FER2013 dataset and a separate system for sentiment driven generation of music artefacts using the revised MIREX like dataset. The emotion classification recognition model classifies faces in 7 different classes: angry, disgust, fear, happy, neutral, sad, surprise. After loading the train and test data the preprocessing function for the model is used on all images to normalize pixels values to a range [-1, 1].

$$x = (x/127.5) - 1$$

Image augmentation is the next step in the preparation of the data: the Keras function *ImageDataGenerator()* is used to apply image shear, rotation, zoom, width shift, height shift and horizontal flip. This makes dataset generation much easier as these steps do not have to be carried out manually using individual functions, also contributing to the systems efficiency by reducing latency. The CNN model with depthwise separable convolutions that is used only works with a minimum image size of 71*71 pixels therefore an important step in the preprocessing of the images is the resizing of the dataset from 48*48 pixels to the minimum allowed of 71*71. The augmented dataset is generated for both train and test samples resulting in 35887 train images and 7178 test images. The FER2013 database does not contain the same number of samples for each class of emotions, this phenomenon is known as class imbalance and can lead to the network not performing optimally. To get around this issue different strategies can be employed: such as upsampling the data, which involves randomly propagating classes with fewer samples in order to obtain more images, however this does not always work and can lead to the overfitting and bad system performance. Another workaround is downsampling the data by diminishing the number of samples for the classes that contain a higher number of images, however the drawback of this method is that it is prone to cause underfitting of the model. The implementation of Class weights is a way to deal with imbalance in the dataset by assigning different weights to the classes based on the number of samples they contain: a class with many images will be assigned a lower weight while a class with fewer images will have its weight set to a higher value. 3 metrics are used for the evaluation of the system: accuracy recall and precision because when there are certain imbalances in the data 'accuracy' may not be a good indicator of performance because even with a high accuracy value the model may underperform. Precision is a metric that quantifies how many correct positive predictions are made and thus it calculates the accuracy for minority classes. The precision of a model for multi class classification is obtained from dividing the sum of true positives across all classes by the sum of true positives and false positives. Recall is a metric which has the aim of quantifying the number of correct positive predictions made from all the positive predictions that could have been achieved. For multi class classification recall is obtained by the sum of true positives in all classes. While precision gives insights into the correct positive predictions out of all positive predictions, recall gives an indication of the missed positive predictions too.

The Keras model Xception is used, this architecture employs the depthwise separable convolutions discussed in the Preliminaries section. Table 1 below shows a comparison of some available models provided by Keras: top 1 and top 5 accuracy refer to a models performance using the ImageNet validation dataset. Depth refers to number of layers with parameters and includes activation layers, batch normalization etc.

Model	Size (MB)	Top1 Accuracy	Top5 Accuracy	Depth	GPU (ms)	Parameter nr
Xception	88	79%	94.5%	81	8.1	22.9M
VGG19	549	71.3%	90%	19	4.4	143.7M
DenseNet201	80	77.3%	93.6%	402	6.7	20.2M
MobileNet	16	70.4%	89.5%	55	3.4	4.3M
ResNet101	171	76.4%	92.8%	209	5.2	44.7M
Incep-tionv3	92	77.9%	93.7%	189	6.9	23.9M

Table 1 – Model comparison

The Xception model was chosen as it is a good compromise between these attributes, with a relatively compact size of 88MB and lower number of parameters of 22.9M compared to models that perform similarly while still outperforming them. The Xception model has a default input size of 299*299 pixels therefore the input shape needs to be addressed by passing the dataset generated previously, converted to 71*71 pixels images and thereafter the top layer needs to be dropped by setting the *include_top* flag to False. Since the Xception model is very large it may not always be possible to train therefore the system has a Trainable parameter which can be turned on or off. In the case it is off, the model uses pre-trained parameters.

If the Trainable parameter is set to True more layers are needed to classify the features, this is achieved by having 2 sets of subnetworks containing dense layers. Care is taken to overcome overfitting by also having 2 regularization layers as demonstrated in the figure 2.

```
def classifier(self, x):
    if not self.trainable:
        x = Dense(dense_1, activation='relu')(x)
        x = Dense(dense_1)(x)
        x = BatchNormalization()(x)
        x = relu(x)
        x = Dropout(rate)(x)

        x = Dense(dense_2, activation='relu')(x)
        x = Dense(dense_2)(x)
        x = BatchNormalization()(x)
        x = relu(x)
        x = Dropout(rate)(x)
```

Figure 2

While training the model may not improve over time and therefore it is monitored during every epoch through a callback function which will stop the training using the *EarlyStopping* function. After training the model successfully this is saved as an HDF5 file however because large files of this type can worsen the latency of the system an additional step is taken. This involves converting the saved model to TFLITE format which converts the saved weights from FLOAT64 to INT8 and reduces the size by a factor of close to 12, in the case of the proposed system reducing the size from 264MB to 22MB making loading the model much quicker. Figure 3 portrays the model diagram [28]
The model achieved good results, with a loss value of 0.1664, an accuracy of 0.9403 and an AUC (area under the curve) of 0.9965. The time takes for training was 7.45 hours or around 1.065 hours per sentiment class.

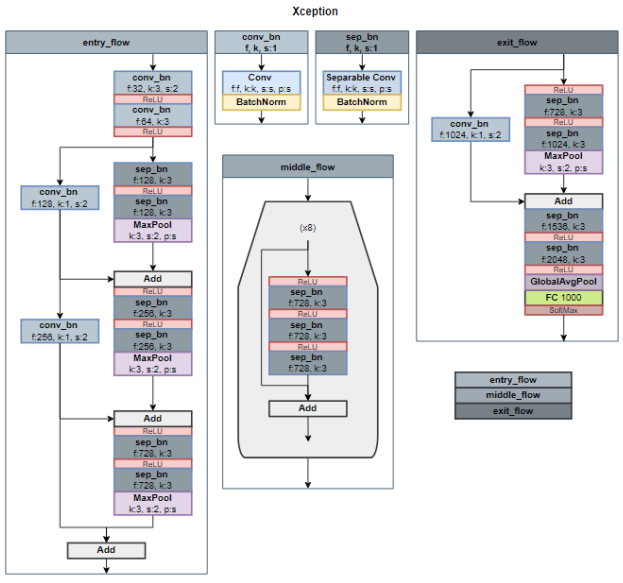

Figure 3

B. Music Generation Model

The music generation model is developed to work in harmony with the emotion detection system however both of these can be utilized as standalone systems. Unlike images, audio files provide data in a sequential form therefore in order to be processed they need to be divided into intervals. This is achieved by having a window which goes over the midi file extracting notes as explained in the Preliminaries section. The model used is a doubly stacked LSTM with a set of batch normalization, dropout and dense layer combinations. Figure 4 shows a visualization of the model extracted with Netron, a software which is able to display models by loading their HD5F or TFLITE files.

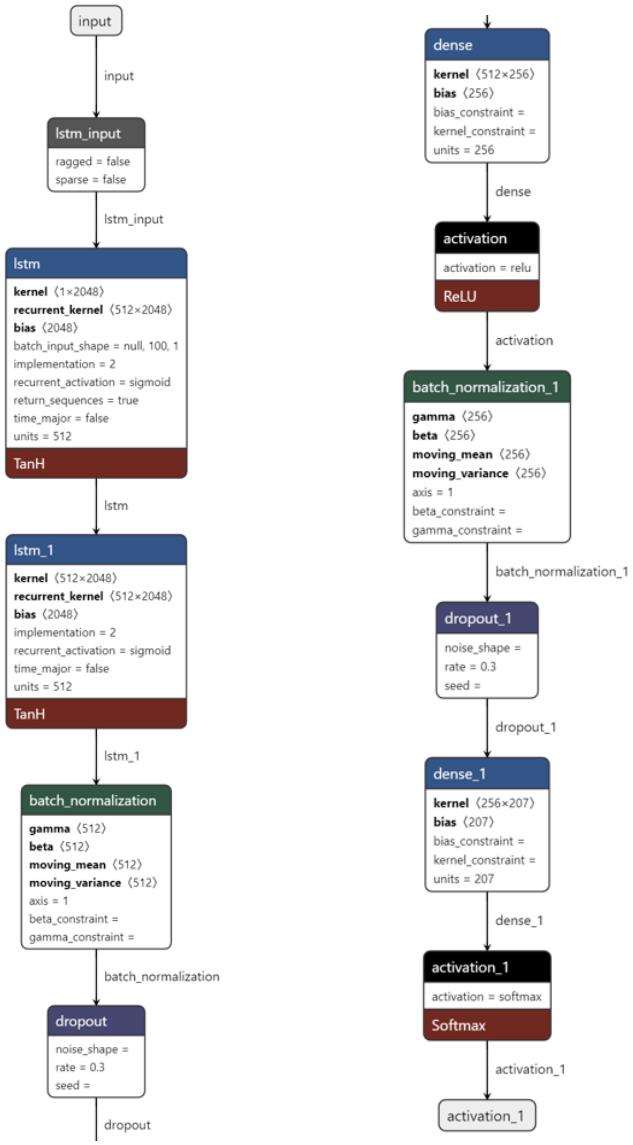

Figure 4 – music generation model (left to right)

More specifically the network is structured in the following manner: an LSTM cell with output dimensionality of 512, followed by an identical cell, a batch normalization layer, a dropout layer set to 0.3, a dense layer with output dimensionality of 256 with activation function ReLU, a batch normalization layer, a dropout layer of 0.3 followed by a dense layer of output shape equal to the length of the input sequence with activation softmax.

In the music generation task, the loss is monitored during the individual epochs, so that the weights can be saved, and callback mode set to *checkpoints*. The number of epochs is set to 200 and the model is fit using the input and output sequences generated after parsing the MIDI files using the MUSIC21 library. Classes of certain emotions are merged into one because of the lack of sufficient data: after conducting unsuccessful experiments the fear, disgust and angry classes were merged into the sad class. The surprise sentiment was merged into the happy class. The training for the music generation has to be subdivided into 3 parts as these are the number of classes of sentiment which are available. The model is trained for the sad, happy and neutral cases and the times for training are 25.03, 31.8 and 21.8 minutes respectively. These figures were obtained by training the model with the following specs: Nvidia RTX 3080 GPU, 32gb RAM, 20-core CPU 6950X @ 4.00GHz.

C. Inference

Once both models are trained the pipeline can be executed at once, this is done using the *Inference* file. This is the process which runs data through the model in order to generate the output. Firstly, the trained TFLITE model with weights saved during the training of the emotion detection system is fetched using the *model_converter* function. The music generation happens in the inference file: here it is passed a parameter containing the sentiment of the expected output. A random sequence from the input sentiment class is picked as a starting point for the prediction of the next 500 notes thus generating the prediction output. At this point, since the data trained in the music generation system was not in a MIDI format but was converted to MUSIC21 object, it cannot yet be synthesized, therefore the generated output is a list of notes which has to be converted back to MIDI format. This step is performed in the *create_midi* function where notes of a sequence are iterated through and extracted as figure 5 demonstrates.

```
for pattern in prediction_output:
    # pattern is a chord
    if ('.' in pattern) or pattern.isdigit():
        notes_in_chord = pattern.split('.')
        notes = []
        for current_note in notes_in_chord:
            new_note = note.Note(int(current_note))
            new_note.storedInstrument = instrument.Piano()
            notes.append(new_note)
        new_chord = chord.Chord(notes)
        new_chord.offset = offset
        output_notes.append(new_chord)
    # pattern is a note
    else:
        new_note = note.Note(pattern)
        new_note.offset = offset
        new_note.storedInstrument = instrument.Piano()
        output_notes.append(new_note)
```

Figure 5 – algorithm for generation of MIDI file from note sequence

The MIDI stream is thereafter written to a file and saved as the final output for the system. In order to invoke the pipeline's inference, the *inference* and *generate* functions are used. The parameter for the inference function is the location of the image that is being fed to the emotion recognition which returns a sentiment. In this step the conversion of emotions also takes place, since certain sentiments have been merged in a single one if statements are used to return the 'sad' emotion when the disgust, fear and angry emotions are encountered. This is then passed to the generate function together with the path where the generated music file is saved, concluding the pipeline.

V. CONCLUSION

This project described a pipeline to generate emotional music artifacts based on the emotion of faces detected by an emotion recognition system. Various techniques apprehended during the academic year are utilized, specifically those from the computational creativity, music informatics, machine learning for visual data analysis and introduction to computer vision modules. The project objectives have been met and the final system comprises 3 files: one for music generation which trains the model for 3 different sentiment classes, one for face emotion recognition which trains the model using a state of the art architecture achieving over 94% accuracy and one for the inference of the system where the elements of the two systems are merged into one so to allow the pipeline to execute efficiently. The emotion of the faces detected affect the sentiment induced by the melody generated and the system is able to generate unique artefacts each time it runs. Methods for making the systems more efficient and improve accuracy have been explored and applied to produce a final system which is refined under these aspects. Audio data is handled and adapted so to work with the TensorFlow library. The datasets have been curated to promote better results and the necessary steps have been taken to correctly label audio files with the correct emotions. Standard machine learning practices have been applied throughout the system, code has been commented and variable naming conventions were followed. This project

demonstrates how through A.I. emotion detection can be applied to sentimental music generation tasks, leaving room for further studies and discoveries in the field of computational creativity. Because of the relatively short time computational creativity was researched for compared to other A.I. fields, many challenges were encountered, and it was difficult to find a right path to follow in order to complete the project, however thanks to the multitude of resources available online nowadays building this system was made somewhat more comprehensible and straight forward.

VI. CRITICAL ANALISYS

The system proposed in this project uses a manually labelled dataset containing music files, however the number of files available is not enough to enable the generation of musical artifacts for all the sentiment classes originally planned. This approach is not the most optimal however due to a lack of available datasets and time constraints the dataset generated is a good compromise in terms of number of samples available and quality of generated artefacts. The latter however can be improved by conducting further research in the field of music generation and paired with the use of a higher sampled data set the quality of the generated melodies can be improved. The complexity of the melodies in regards to musical characteristics is deemed acceptable in this project but compared to human generated songs it does incite the same level of emotional response. This could be improved if more resources are made available for the generation of an appropriate dataset. Midi files are not further curated, however one advantage of MIDI over other audio files is that they can be synthesized in different ways and played by different instruments thus improving the emotion they arouse in the listener. Because of the nature of the generated result, it is hard to evaluate the melodies using conventional metrics. Computational creativity consists of generating artefacts and evaluating art is not always a straightforward process however there are some methods of evaluating the outputs generated by the system and results are discussed in this section. The system was used to generate 6 samples (2 for each sentiment class) and 6 listeners were asked to evaluate how closely they feel that the melody they listened to invokes the intended sentiment. For the sad class 6/12 people responded that the melody provokes the sadness, 2/12 responded that the melody somewhat provokes sadness and 4/12 responded that the melody does not provoke sadness. For the happy class the respective figures are 9/12, 2/12 and 1/12. The neutral class results were 4/12, 6/12 and 2/10.

VII. FURTHER WORK

Further work use this system may involve changing the architecture to use the MobileNET model and create an app which can work efficiently on mobile devices. It could also be possible to make it work with videos and therefore in the wild data if trained with a different dataset. The reason sometimes songs were not recognized to belong to the correct emotion is be cause of the way the midi files are interpreted. Changing the synthesizer and playing the notes using different instruments can have a big impact on how the output is perceived. This type of post processing however belongs more to the musical arts field of study and is not very necessarily related to the use of artificial intelligence in this project.

VII. ACKNOWLEDGMENTS

Firstly, I would like to thank my project supervisor, Professor Marcus Thomas Pearce for providing me guidance early in the project and giving me the freedom and opportunity to approach my project in an analytical and sensible manner, his understanding and professionalism. I want to also thank Queen Mary University of London and my lecturers for providing and teaching the material required in order to carry out this project successfully as well as the resources and support needed. I have thoroughly enjoyed carrying out this work and I am thankful for the opportunities I was given to evolve as a Computer scientist. I also want to thank my family and friends who have supported me morally and emotionally during the hardships of this project and helped me deal with pressure.

REFERENCES

[1] Y.-I. Tian, T. Kanade, and J. F. Cohn, “Recognizing action units for facial expression analysis,” IEEE Transactions on pattern analysis and machine intelligence, vol. 23, no. 2, pp. 97–115, 2001.

[2] C. Darwin and P. Prodger, The expression of the emotions in man and animals. Oxford University Press, USA, 1998.

[3] P. Ekman and W. V. Friesen, “Constants across cultures in the face and emotion.” Journal of personality and social psychology, vol. 17, no. 2, pp. 124–129, 1971.

[4] R. E. Jack, O. G. Garrod, H. Yu, R. Caldara, and P. G. Schyns, “Facial expressions of emotion are not culturally universal,” Proceedings of the National Academy of Sciences, vol. 109, no. 19, pp. 7241–7244, 2012.

[5] P. Ekman, “Facial action coding system (facs),” A human face, 2002.

[6] E. Rosenbaum, Y. Mann and Google Creative Lab. “Experiments with Google: Giorgio Cam”, 2016

[7] Rigg, M. G. (1940). Speed as a determiner of musical mood. Journal of Experimental Psychology, 27.

[8] Parncutt, R. (2014). The emotional connotations of major versus minor tonality: One or more origins? Musicae Scientiae, 18.

[9] M. Matsugu, K. Mori, Y. Mitari, and Y. Kaneda, “Subject independent facial expression recognition with robust face detection using a convolutional neural network,” Neural Networks, vol. 16, no. 5-6, pp. 555–559, 2003.

[10] B. Sun, L. Li, G. Zhou, X. Wu, J. He, L. Yu, D. Li, and Q. Wei, “Combining multimodal features within a fusion network for emotion recognition in the wild,” in Proceedings of the 2015 ACM on International Conference on Multimodal Interaction.

[11] B. Sun, L. Li, G. Zhou, and J. He, “Facial expression recognition in the wild based on multimodal texture features,” Journal of Electronic Imaging, vol. 25, no. 6, p. 061407, 2016.
[12] Christopher Olah. “Understanding LSTM Networks.” https://colah.github.io/posts/2015-08-Understanding-LSTMs/ (2015).
[13] Eck, D., & Schmidhuber, J. (2002). Finding temporal structure in music: Blues improvisation with LSTM recurrent networks. In Neural networks for signal processing - proceedings of the ieee workshop (Vol. 2002-Janua).
[14] Boulanger-Lewandowski, N., Bengio, Y., & Vincent, P. (2012). Modeling temporal depen12 dencies in high-dimensional sequences: Application to polyphonic music generation and transcription. arXiv preprint arXiv:1206.6392.
[15] Hadjeres, G., & Nielsen, F. (2017). Interactive music generation with positional constraints using anticipation-rnns. arXiv preprint arXiv:1709.06404.
[16] Ferreira, L. N., & Whitehead, J. (2019). Learning to generate music with sentiment. In Proceedings of the 20th international society for music information retrieval conference, ismir 2019.
[17] J. M. Susskind, A. K. Anderson, and G. E. Hinton, “The toronto face database,” Department of Computer Science, University of Toronto, Toronto, ON, Canada, Tech. Rep, vol. 3, 2010.
[18] I. J. Goodfellow, D. Erhan, P. L. Carrier, A. Courville, M. Mirza, B. Hamner, W. Cukierski, Y. Tang, D. Thaler, D.-H. Lee et al., “Challenges in representation learning: A report on three machine learning contests,” in International Conference on Neural Information Processing. Springer, 2013, pp. 117–124.
[19] “Static facial expression analysis in tough conditions: Data, evaluation protocol and benchmark,” in Computer Vision Workshops (ICCV Workshops), 2011 IEEE International Conference on. IEEE, 2011, pp. 2106–2112.
[20] Thierry Bertin-Mahieux, Daniel P. W. Ellis, Brian Whitman, and Paul Lamere. "The Million Song Dataset". In *Proceedings of the 12th International Society for Music Information Retrieval Conference*, pages 591–596, 2011.
[21] Colin Raffel. "Learning-Based Methods for Comparing Sequences, with Applications to Audio-to-MIDI Alignment and Matching". *PhD Thesis*, 2016.
[22] Panda R., Malheiro R., Rocha B., Oliveira A. & Paiva R. P. (2013). "*Multi-Modal Music Emotion Recognition: A New Dataset, Methodology and Comparative Analysis*". 10th International Symposium on Computer Music Multidisciplinary Research -- CMMR'2013, Marseille, France.
[23] Sargam Modak, (2017), ‘*StackOverflow: Normalizing to [0,1] or [-1,1]*’, Accessed Aug 2022: https://bit.ly/3PZ9pFe.
[24] Cuthbert, Michael Scott and Christopher Ariza. "music21: A Toolkit for Computer-Aided Musicology and Symbolic Music Data." in J. Stephen Downie and Remco C. Veltkamp (Eds.). 11th International Society for Music Information Retrieval Conference (ISMIR 2010), August 9-13, 2010, Utrecht, Netherlands. pp. 637-642.
[25] https://openaccess.thecvf.com/content_cvpr_2017/papers/Chollet_Xception_Deep_Learning_CVPR_2017_paper.pdf
[26] https://arxiv.org/pdf/1704.04861.pdf
[27] https://arxiv.org/pdf/1409.4842.pdf
[28] MLT MachineLearningtokyo.com “CNN ARCHITECTURES: XCEPTION”, (2020), accessed August 2022