MUSIC GENERATION FOR SCENE EMOTION USING GENERATIVE AND CNN MODEL

D.I.D.D. Jayawardena

168285U

Degree of Master of Science in Artificial Intelligence

Department of Computational Mathematics

University of Moratuwa
Sri Lanka

March 2019

MUSIC GENERATION FOR SCENE EMOTION USING GENERATIVE AND CNN MODEL

D.I.D.D. Jayawardena

168285U

Thesis submitted in partial fulfilment of the requirements for the degree of Master of Science in Artificial Intelligence

Department of Computational Mathematics

University of Moratuwa

Sri Lanka

March 2019

Declaration

I declare that this is my own work and this thesis does not incorporate without acknowledgement any material previously submitted for a Degree or Diploma in any other University or institute of higher learning and to the best of my knowledge and belief it does not contain any material previously published or written by another person except where the acknowledgement is made in the text.

Also, I hereby grant to University of Moratuwa the non-exclusive right to reproduce and distribute my thesis, in whole or in part in print, electronic or other medium. I retain the right to use this content in whole or part in future works (such as articles or books).

Name of Student:
D.I.D.D.Jayawardena.

Signature of Student: Date:

The above candidate has carried out research for the Master's Dissertation under my supervision.

Name of Supervisor:
Dr. Subha Fernando.

Signature of Supervisor: Date:

Abstract

Generate music using emotional semantics of an image is quiet challenging task due to the complexity of extracting emotional features of an image and generate music according to the emotion.This paper proposes an enhanced deep neural network backed by Generative adversarial network for scene emotion categorization and LSTM based music generator for music generation. In developed system system functions in three parts.Initially we have generated fake images which more looks like real images using the generator of generative adversarial network which will help to enrich the dataset and increase the size of the dataset.Our dataset contains mainly three emotion categories (Happy,Angry,Sad). Second part of the system is image classifier developed using convolutional neural network which is trained using enhanced image dataset of scene emotions.Image classifier helps to identify the probabilities of the input scene which fed in to music generator for creation of training music dataset for each uploaded scene.Third and the last part of the system is the music generator which is developed using convolutional neural network with Long short term memory model.With the use of LSTM model developed deep neural network model got the capability of remember and predict next step.MIDI dataset from raw music files of songs created for each category to train the music generator. Since music composing is more human centric task,best way to evaluate the system is using musicians.So we have tested the system with two musicians and single listener.And also we have compare the image classifier using dataset which contains GAN generated images and without GAN generated images. After improving the dataset using generated images by GAN,we were able to achieve 80% of categorical accuracy and 85% of validation accuracy in image classification.Based on the evaluation done by musicians on generated sounds more than 50% of the sounds were in good quality and they have confirmed the musics were appealing to hear.

Acknowledgements

It is quiet a long journey to learn and successfully complete a research.So lot of people from academic side and family end helped to succeed in this work.First I would like to thank my supervisor for giving me full freedom to do the work as my way and guiding me when I need help. Then I have thank my parents and sister helping me and encouraging me to go for success.My friends were always with me to help when I really need motivation to work on research with the hectic work life in this country.

Contents

List of Figures

List of Tables

List of Abbreviations

CNN	**C**onvolution **N**eural **N**etwork
GAN	**G**enerative **A**dverserial **N**etwork
WGAN	**W**asserstein **G**enerative **A**dverserial **N**etwork
LSTM	**L**ong **S**hort **T**erm **M**emory
RNN	**R**ecurrent **N**eural **N**etwork
RNN-RBM	**R**ecurrent **N**eural **N**etwork **R**estricted **B**oltsman **M**achine
ASC	**A**coustic **S**cene **C**lassification
ACGAN	**A**uxiliary **G**enerative **A**dverserial **N**etwork

Chapter 1

Introduction

1.1 Prolegomena

Music is coming from prehistoric era of humans as entertainment medium as well as communication medium.Music has direct relation with our biologies as well as our social interactions which lead to evolution of modern human mind[1].Due to this reason people use music to express emotional feelings via music and also it is very much closer to human mind and heart.So in modern world music became very powerful and huge industry.Many researchers were involved in making novel systems for increasing business requirements of this huge entertainment industry.Another concern of many researchers and the industry is how to build more realistic machines which could imitate the human behaviors.So they are more curious about the fact of could machines be creative.To achieve this researchers are trying to computationally model existing man made models which were used in day today works.Some of these models were golden ratio (used in graphic designing), color theory, Emotional models.To model these computationally people use multi agent systems,genetic algorithms,deep convolution networks(CNN)[2, 3, 4].In this research project main intention is to generate a model which is capable of classifying a scenery in to appropriate emotional category(description) and using that information to generate novel music that did not exist accordingly.As per the requirement,plan is to to use generative model and CNN model conditioned with long short term memory(LSTM) which which are capable of generating new knowledge that did not exist.

1.2 Objectives

The aim of this study is to build an entertainment application which could generate music by identifying the emotional qualities of the scenery which is provided as input.

- Critical study on generative models and CNN models and its applicability in this research topic.
- Critical study on current solutions which are available in the market for this issue.
- Design a system and algorithm to handle the research problem and develop the full functional application.
- Evaluate the system with generated data by the system.

- Producing the final documentation.

1.3 Background and Motivation

image classification and vision based information retrieval are the major areas in vision based systems.Lots of work going on to develop better model to classify the images and information retrieval.Most of the researchers were using traditional basic models to design systems.one of the major model used by them is color descriptors.One of the work done by Sande explains how color descriptors were used to increase the power of illumination invariance and discrimination[5].Szabolcs sergyan's work also describe how color descriptors could be used to analyze images.As author mention color is an important factor in content base image retrieval. And also author describes which are the possible color descriptors which could use to classify images.And also describes which color descriptor could be used in which color spaces[6].Visual auditory associations to music was the main focus in work done by Xiaoying Wu and his team.They have mapped primitive visual qualities like contours,colors and texture to musical elements such as pitch,duration and chords. According to them the results that they got were impressive but the draw back was only small sound clips were able to generate.Generating complete piece of music is more challenging task[4].Hye-Rin and his team also worked on recognizing image emotions through deep neural networks.They describes a feed forward network model and the outputs were valence-arousal values which express the emotion and they used image sentiments of background, color models and objects to decide the emotion category of an input image.But they have faced major issue in their work. That is object recognition accuracy directly impact on the performance of the emotion recognition model[7].So as these works researchers were working more towards the basic color theory concepts to retrieve image information and categorize them. Bolei Zhou in his work also tried to classify images using convolutional neural network(CNN). They used scene-centric database called palces with over 7 million labeled pictuures of scenes whichis more diversified dataset than other datasets.They have shown that the differences in internal representation of object-centric and scene-centric networks using the visualization of CNN layers[8]. Most of the image categorization models fails due to lack of dataset which are relevant to the required categories.Most novel way of solving this kind of problem is models like generative adversarial networks(GANs) where we use set of labeled images and using the generator and discriminator model to generate non existed data which could be predicted to be appear in future occurrences. As mentioned above Xinyue Zhu and the team used GAN with data augmentation to classify emotional images.They used convolution neural network(CNN) model to classify the emotions and used GAN to generate lacking images to provide supplement to existing dataset[9]. Matic and

Dragon worked on genetic algorithms to make computational model to generate music.Position based representation of rhythm and pitches used by them to encode music composition.And they were able to generate beautiful compositions which is one of their objectives.And also author mentioned that rhythm was meaningful and compositions were pleasant to hear[2].Using wave forms as genetic codes Manzolli and the team generated sounds. So by crossover and mutation done to the waveform lead to create new waveform[3]. As this way music and sound generation task taken seriously by many researchers.

1.4 Problem in brief

Generating music on a given scenery using it's embedded qualities is more human centric creative task. Most of the tasks related to creativity such as designing, drawing, musical activities were human centric. And there are lot of discussions and researches going on to bring these human centric capabilities to machines. Identifying an emotion embedded in an image or scenery is quite challenging task for a machine. And there is no exact rules defined in this matter. And using those ingredients included in the image and generating music accordingly involve creative thinking which also does not have hard and fast rules. As a conclusion both these tasks need more human thinking and creativity.

1.5 Proposed solution

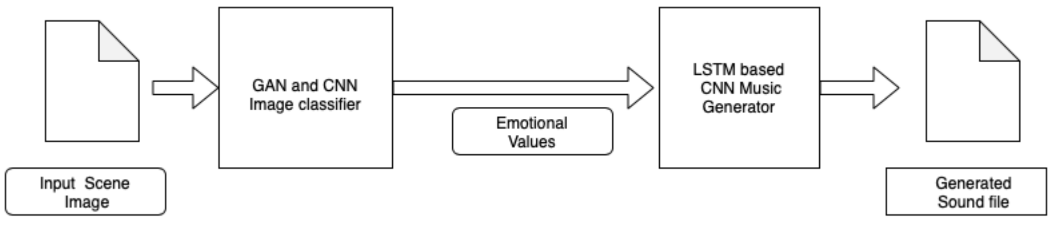


FIGURE 1.1: Overview of proposed solution.

We have hypothesized that the problem of generating music for scene emotion could be solved using combination of generative adversarial networks with deep convolution neural networks for image sentiment analysis and generate music without intervention of human other than the provided scene image. In this project system is developed basically in two phases. First phase is to identify the emotional categories of the input scenery image. Expected categories are happy, sad, angry. For this task conditional generative adversarial network implementation and deep CNN used. To train the network we use freely available data set with emotional description embedded to it. And the input to the system will be a random image of a scenery. Since we need huge computing power to process the images this implementation require cloud instance like amazon or google.

Second part will be using the generated probability output from first phase and generate music accordingly. For this purpose we use LSTM based deep convolution neural network solution. To train the model using a self prepared midi data set. So the final output of this phase is set of novel music clips. Finally the input and music clips were integrated as one output as an application. Users of the system will be Administrator, who is responsible for the labeled image database and labeled sounds database and application users.

1.6 Resource requirements

Following are the resource requirements for this work,

- Google cloud based instance.
- Computer with high processing power.
- Several libraries such as tesorflow, keras,numpy.
- python programming language based tools(Ipython notebook).

1.7 Structure of the thesis

This thesis has been structured with 8 chapters. Chapter 1 gave an overall introduction the project. Chapter 2 provides a critical review of the developments and issues in the area of music generation for scene emotion by defining the research problem and identification of technologies. Chapter 3 is on technology adapted to building the GAN and CNN based solution for Scene emotion detection and music generation. Chapter 4 presents our approach to music generation for scene emotion using GANs and deep CNN. Chapter5 Design of the solution. Chapter 6 Implementation of the designed system. And in Chapter 7 Evaluation of the system created after training the algorithm. Finally Chapter 8 concludes the thesis with a note on the possible further work.

1.8 Summary

This chapter provided an introduction the entire project. For this purpose, we have presented our research problem, objectives, technlogy dapoted, proposed solution and resource requirements. Next chapter provides a detailed critcal review of Music generation based on scene emotion.

Chapter 2

Issues and Challenges

2.1 Introduction

Chapter 1 gave an introduction to overall project. This chapter presents a critical review of literature on music generation for scene emotion. Here we formulate our research problem and highlight the technology adopted towards a solution. In doing so, this chapter has been structured with four sections, namely, gestation, major developments in image sentiment identification, major developments in music generation and problem definition.

2.2 Gestation of music generation for scene emotion

Image Classification and vision based information retrieval are the major areas in visioned base systems. Lots of work going on to develop better model to classify images and information retrieval.To address these issues people are using more basic models and make systems.Most of them were using color based classification models.One of the work done by Szabolcs Sergyan explains how color descriptors could be used to analyze images and which descriptors could be used according to the color space.As author mentioned the best results could be achieved by using the moments and color coherence vectors. And also he mentioned that for similarity retrieval tasks histogram based classification does not work well [6].Xiaoying Wu and the team worked on visual auditory associations to music generation.They have implemented several methods to convert visual features to music elements like pitch,duration and code.They have used 3 steps to compose music.Those steps were partitioning step,sequencing step and mapping step.In first step they partition the image features and in second step decide the sequence of the partitioned pieces.They used contour, color and texture to partition images.After that in mapping step they convert each piece to music note.Melodic anchoring principals used in mapping the notes together because if the same sequence of partitioning will not match with music sequencing.As per the authors they were able to generate pleasant music but the quality of the music generated completely depend on quality of the images used.They also suggest to use tonal hierarchies and emotional models to generate music in future works [4].The work done by Hye-Rin and team on image emotion detection.They have used two features in emotion detection.Those were the objects in an image and the what background the object is in.Based on these kind of semantics they developed emotion based feed forward deep neural network. The output generated by the model was in 2 dimensional space which contains valance and arousal values.As author mentioned there were huge impact on prediction of emotion with the object detection if the system identifies the object as a wrong one then the emotion prediction completely get mess up [7].

2.3 Major developments in music generation

Music generation and composing is hot topics lot of researchers interested in.Graves in his work used recurrent nural network with LSTM model to generate more realistic hand writting.Chung in his work extended Graves recurrent neural network(RNN) model with long short term memory by adding gated recurrent unit with LSTM for polyphonic music generation and speech signal modeling.Both the LSTM units and GRU units capable of keeping the existing content and add new content preserving old one which is not possible in recurrent units.As author mentioned both LSTM and GRU based systems perform well than traditional models [10, 11].

Boulanger-lewandowski introduced a Recurrent neural network restricted boltzman machine(RNN-RBM) model which is a probabilistic approach.By using RTRBM which is energy based model they were able to predict next step easily for high dimensional objects.With that conditional distribution of previous time step and next time step is predicted.This helped author to discover temporal dependencies in a high space model.[12].

NatashaJaques and the team worked on a research about fine tuning sequence generation model with KL-control which was named as Sequence Tutor.Author has created off policy reinforcement learning (RL) model from KL-control.This model is used for generation novel music melodies and computational molecular generation and represented the effectiveness.The author′s intention was to tune some properties of the model without interfearing the original probability distribution of the data. so they proposed this sequence tutor solution.As author mentioned they were able to generate quality sequences as expected [13].

Jonatas and the team explains how waveforms could be used with genetic algorithms to generate new music. Further they describe how wave forms could be used as genetic codes.They have used method called ESSynth which integrates the Mathematical Approximation Theory to the Genetic Algorithms.With this wavetable synthesizer is used to instruct the wavetable to play the sound pattern.As author mentioned quality of the generated waveforms depend on waveforms of target set[3].

Ramanto and the team used markov chain based procedural music generation method which is a stochastic model used in modeling the component of music composition.They used Tellegen-Watson-Clark circumplex model to represent mood in relation to music.By their work they were able to generate music for given mood[14].

Interactive genetic algorithm used by Sung Bae Cho to generate music as well.In their work users could add manual inputs to the fitness function. But the issue was every user do not have music knowledge to interact with the system.Since image retrieval process done using content-based image retrieval the image database creation.The author used two ways to music retrieval,One way is using IGA and the other one is using a query from music database[15].

Saber Malekzadeh and the team used auxiliary generative adverserial deep neural network(ACGAN) for categorical music generation.They used hybrid architecture which had different kind of layers of neural networks.First they have transformed the input data from time domain to time frequency domain using Short-Time Fourier Transform(STFT).Using those time frequency domains the GAN is trained.After that generated frequency domains converted using inverse STFT. They were able to achieve 75.6% of sucess rate in generated music after given to musicians to evaluate the quality [16].

Sandr Dielema and team used auto-regressive discrete auto encoders(ADAs) in their attempt to model music using raw audio domain.This ADAs enabled the capability of capturing the long range correlations in wave forms and they were able to generate piano music by using raw audio domain[17].

Segment concatination methods and hash learning algorithms were the inspirational models for the novel approach introduced by the Kevin Joslyn and the team.They used an approach called Deep segmentation hash learning.Which is the first end to end segment hash learning method for music generation.They used pair wise training system to train the model.In this case algorithm can say that a segment is suitable to compose as the next segment of the composing music by matching similar occurance in the dataset.They were able to generate original and enjoyable music as well[18].

Stefan Lattner and team used convolutional restricted boltzman machine(C-RBM) as a generative model to generate polyphonic music. With their model they could control the higher -level self similarity structure, The meter and the tonal properties of the generated musical piece[19].

Jay A Henning proposed an extention of the VAE framework.They combined RNN with VAE to model sequential data which is like LSTM[20].Kratarth Goel and the team proposed a system which is a combination of a RNN and a deep belief network.By using their system they were able to represent more complex data than an RBM[21].

Mason Bretan and his team worked on a project where unit selection and concatenation used for music generation.They first created deep autoencoder which could be used to encode the musical input and using the encoded data library recreate the input structure.After that they created deep structured semantic model with LSTM to predict next units.Where deep structured semantic model is used to get sementic relevance score and concatenation cost is calculated by the LSTM.These scores used to rank the units of the library.The useful part of this solution is that directly previously composed music could be used and concatenate.But in here finding the exact unit length was challenging.As author explain the evaluation process also challenging due to musical variance and still the concatenated music is valid music.So author suppose subjective listening test for evaluation[22].

Deep Artificial composer (DAC) which is capable of composing monophonic melodies with compositional structures created by Florian Colombo and the team.They were able to compose music which were similar to human composers and also consistent.DAC was implemented using multi-layer (deep) neural networks with LSTM units.They have converted symbolic notations to midi notes using music21 python library before entering to the system.As per the author mentioned they were able to achieve high accuracy in average predictive performance of the duration model.The values were between 80% to 85%.And also the author has shown the capability of DAC to learn similarities and differences in music styles.Novelty measurements to classify the styles of melodies were used in evaluation process[23].

Huanru Henry Mao and the team developed a model called DeepJ where the users or composers could add inputs to system while composing the melodies. In their work they have used biaxial LSTM architecture with some changes to the representation using data augmentation with music dynamics.They trained the model with three styles of music (baroque, classical and romantic) using midi music files.They have evaluated the model in quality by using subjective experiment and style by using 20 individuals with music back ground.By using this model the author was able to solve style consistency issue as well.In this model the problem faced was lack of long term structure.So the author suggest to use adversarial methods or combining reinforcement learning methods[24].

Jambot is also a LSTM based polyphonic music generator developed by Gino Brunner and the team which designed to generate music in two steps.The first step is use of chord LSTM to predict a chord progression based on a chord embedding and then the second step is to using generated code progression generate polyphonic music using LSTM.They used Lakh MIDI Dataset for training the model.As author mentioned the model itself learned music theory related principals by observing the dataset.Since this model is simple and they use midi data it could provide fast results[25].

Vasanth Kalingeri and team also worked on music generation using LSTM based deep learning techniques.They used raw audio data with several LSTM architectures to generate music.Not like using midi data from raw audio generating musiic is challenging task because you are not using data related to music structures.They have used Multilayer LSTM where LSTMs stacked to store more features in every time step and LSTMs in bi-linear alignment.Such that several LSTM aproaches used in generating music.As per the author bilinear architecture and LSTM with 2D convolution layers generated good outputs.Author also suggest to use adversarial networks which could be used to model the lose function by the algorithm itself[26].

Another neural network model(LSTM-RTRBM) by Qi Lyu and team were able to compose polyphonic music.In their approach also they used LSTM for memorizing and retrieving important history information.They also integrated Restricted Boltzmann Machine which could perform in high dimensional data modeling. Since the polyphonic music sequence having high dimensions it is challenging to predict next valu step by using just LSTM based RNN.So to get rid of this issue energy based model could be used.In his work author used RTRBM which is capable of log the likelihood of a given configuration.As per author they were able to achieve state-of-the-art results on various datasets[27].

Another model suggested by Hang chu and team using hierarchical Recurrent Neural Network to compose pop music.They encode the prior knowledge in layers and the structure of the hierarchy.They have used scale type generation which allowed the model to pick regularities in the music.They have encode the melody with which key is being pressed and how long it was pressed in each time step. A recurrent neural network is used to generate key conditioned on scale and using that second recurrent neural network generate the duration of key pressed.They have used midi man dataset in training process. As the author mentioned they were able to generate multi track music and the model perform well.So in our work we are providing fully automated deep CNN and LSTM based solution for music generation[28]. In our work by using above mentioned solutions we are using LSTM based deep neural network architecture to generate music.

2.4 Major development in image sentiment identification

Emotion of an image can be evoked by many factors.For the emotion prediction problem many researchers used various types of color statistics of art and psychological features.Machajdi proposed image classifiaction system which used psychology and art theory based features.Ltten's color contrast and rule of third are some of these features[29].

Zhou and the team also proposed the same principal for emotion extraction of an image.They have proposed new new mechanism to compare the density (data concentration) and diversity (variability of appearance) of a dataset.They used densities of nearest neighbors of an image in dataset to compare with other dataset and used a measure inspired by Simpson index for diversity calculation.This mechanism helps to identify the dataset biases which will help in making making more generalize classification model [8].Similarly Xin lu computed the shape based features in natural images[30]. Chen used CNN to achieve better performance in classification problem[31].

Yongli Zhu and team presents work related to power grid disturbace classification by using deep learning.Image embedding technique called Gramian Angular field is used by the author to transform each time of event data to a 2 dimensional image for learning.And they have achieved good results than other methods in power system transient disturbance classification[32].

Arindam Das and the team used a region based deep convolutional neural network framework for document stucture learning and theey were able to achieve accuracy if 92.21% on the dataset of RVL-CDIP[33].Roshan Gopalakrishnan and the team used neuromorphic learning algorithms in classification problem.They trained a deep learning network using CALTECH101 dataset and a collapsed version of neuromorphic CALTECH101 dataset[34].

2.4.1 Problems In Image Classification

In many classification problems the main issue researchers come up is insufficient data to train the system in accurate and make the classifier robust one.So many researchers working on finding proper solution for this issue.One of the solution is to increase the training data set set but in many cases practically it really impossible to find sufficient data.By having this issue people came up with the idea of data augmentation.So in this section we review some of the provided solutions for data augmentation.Alhussein fawzi and his team suggested an adaptive data augmentation method for image classification.As per their proposal,they use new automatic and adaptive algorithm for choosing most appropriate transformation of each sample data.In detail they try to find small transformation in data that could lead to maximal classification error. In finding this they used trust region optimization strategy and they have used sample of 5000 images from MINST hand written digit dataset to train a CNN which has 3 layers.They have critically compared the system with other systems and according to the evaluation the new system perform more accurately.In case of more complex images than hand written images,we need more accurate way of image classification[35].

Wu Ren and the team presents state of art image recognition system.Another issue of in image classification is if the size of the model is higher more computational power is required.As a result author used dedicated super computer in classification problem. Even though they have more computational power they have the issue previously mentioned. because if you have less training data the network will over fit to the data. So still the data augmentation mechanism required for them to train the system. They have used ten thousand times larger dataset in training the system. The author tries to improve the capability of finding important features in classifying objects rather than other artifacts in the image.So they have used several data augmentation mechanisms to achieve this.Author used color casting to alter the RGB channel intensities of training data. And also author has used vignetting effect of images and Lens distortion to augment the data.Other than that author used high resolution images in classifying which helped his identifying small objects in images.To use this kind of approach we need higher computational power and if the dataset differ from object classification the approach may not suitable[36].

Saining Xie and his team introduced a solution for above mentioned sentiment extraction issue.Their proposed new framework named as hyper-class augmented and regularized deep learning or Fine-grained Image Classification.They propose new data augmentation method.where they used fine-grained data to identifying inherent and easily annotated hyper-classes(easily annotated inherent attributes) and collecting huge amount of similar images with same hyper class labeling. And also they have proposed regularization technique to improve the capability of handling the generalization.They were able to achieve 86% accuracy in classifying task with the new approach.Even though this hyper class approach perform well still the sentiment extraction related to lighting and embedded feeling is quiet challenging[37].

The work done by Taylor and Nitschke also used geometric and photo-metric methods of data augmentation to address the small dataset for deep learning issue. They used geometric methods like flipping,rotating and cropping techniques to augment the data and photo-metric methods like color jittering, edge enhancement and fancy PCA.So this lead them to identify lighting related and color related changes in images.As per the authors by all these augmentation techniques the accuracy of classification improved.And also they mentioned that geometric method outperformed photo metric methods . As per them greatest performance acquired by the using cropping method.By this work also it proves increasing the dataset using data augmentation could improve the performance of the deep learning tasks[38].

Zhun Zhong and the team also worked on a project where data augmentation is used to improve CNN performance but their model used for object identification tasks.What they have done was randomly occludes an arbitrary regions region of the training image in every iteration. Which helps to reduce the risk of over-fitting the model. By using their model they were able to increase the performance of the object detection and person re-identification[39].

Data wrapping and synthetic over sampling used by Sebastien C. Wong and his team to achieve data augmentation in data space and feature space respectively. Data augmentation of data space achieved by method called elastic deformation other than the existing transformation methods.Synthetic Minority Over-Sampling Technique (SMOTE) was used by the author to achieve feature space data augmentation.As per the authors both techniques could be used to improve performance of the system.If we required to do label preserving data space transformation could be used and in other cases feature space transformation is suitable[40].

By using above mentioned traditional data augmentation techniques we could only augment the data for certain level.But if generative models used to do data augmentation broader set of augmentation could be achieved.Antreas Antoniou and the team used date augmentation generative adverserial networks(DAGAN) to generate augmented data.As the authors mentioned this method is more powerful to generate data for unseen classes and also really help full in law data setup[41].

To achieve segmentation in heterogeneous and limited amount of data set in medical imaging solutions, Mina Rezaei and team propose conditional GAN based solution.In their model they use descriminator of GAN to descriminate segmentation maps generated by the segmentation CNN.As per the authors they were able to provide more affective brain tumor segmentation and survival day prediction solution using conditional GAN[42].

Hojjat Salehinejad and his team also used GAN based solution for pathology image classification task.In this case they used GAN as a data augmentation method as well as missing or lacking data set generator. The author used DC-GAN for chest X-ray images generation and used deep convolutional neural network for image classification.As author mentioned using synthesized images they were able to get more generalized performance of unseen data classification and get diversified data set other than the data augmentation.Since GAN can generate unseen data it helps to balance the data set and improve the quality of training data set.Work done by Jelmer M. Wolterink and team also confirms the capabilty of improving the bio medical image analysis and segmentation by using generative adverserial networks[43].

Another bio medical research done by Xin Yi and team used categorical generative adversarial network for detection of skin cancer called Melanoma.Which they used to automatic feature extraction in dermoscopy images under supervised and semi supervised environment.And also they were able to generate real world like data by using the GAN.In normal generative adverserial networks there is an issue of descriminator becoming more powerful in that case the generator fails to learn any more. So to avoid this issue WGAN can be used where Wasserstein distance is used to calculate the distance between generated dirstribution and sample real distribution.As per the user model provide good result than using auto encoders and hand crafted features.So as per many researchers mentioned the novel and more powerful way of augmentation is using a GAN based solution. In our work we use Wasserstein generative adversarial network to create rich dataset to train classifier which could work with different unidentified scene before[44].

2.5 Problem definition

Generating music on a given scenery using it's embedded qualities is more human centric creative task. Most of the tasks related to creativity such as designing, drawing, musical activities were human centric. And there are lot of discussions and researches going on to bring these human centric capabilities to machines. Identifying an emotion embedded in an image or scenery is quite challenging task for a machine. And there is no exact rules defined in this matter. And using those ingredients included in the image and generating music accordingly involve creative thinking which also does not have hard and fast rules. As mentioned earlier in this chapter lots of work going on to bring these human centric capabilities to the machines. Still there is lot of gaps in generated models to achieve this task. Generating a model that can manage a situation that was unknown earlier is quiet challenging. Extraction of the semantics in a scenery is require more accurate model than extraction of semantics in a face image. So the model should be trained with more data. Sound generation models should also be more accurate to generate music according to the emotional input. So a better model should be designed to address this kind of matter.

2.6 Summary

In this chapter we have described the problem what we are going to address in detailed manner and we have analyzed the previous works critically to support our problem. Brief history on related researches, and the major trends in music generation and image classification is described in detail.

Chapter 3

Technology For Scene Emotion Detection And Music Generation

3.1 Introduction

In Chapter 2 detailed introduction to the problem domain is provided. This section will provide more information about the technologies which could be used to solve the problem. There are many technologies which we could use but the accuracy of the algorithm needs to be taken in to account when deciding which technology must be used.So main technologies focused on this chapter were deep convolutional neural networks,Generative adversarial networks and long short term memory(LSTM) based music generation.Other than those main technologies further information about programming languages, libraries provided.

3.2 Convolution Neural Networks(CNN)

Main Technology used in this study is convolutional neural networks. So it is important to understand basic concepts related to convolutional neural networks.The natural inspiration of convolutional neural networks comes from visual cortex.Which has small regions of cells that are sensitive to selected regions of the visual field.In a CNN input image pass through several layers and finally get a single classified class or a vector of probabilities relative to each class.Such layers are convolutional,nonlinear,pooling and fully connected layer.

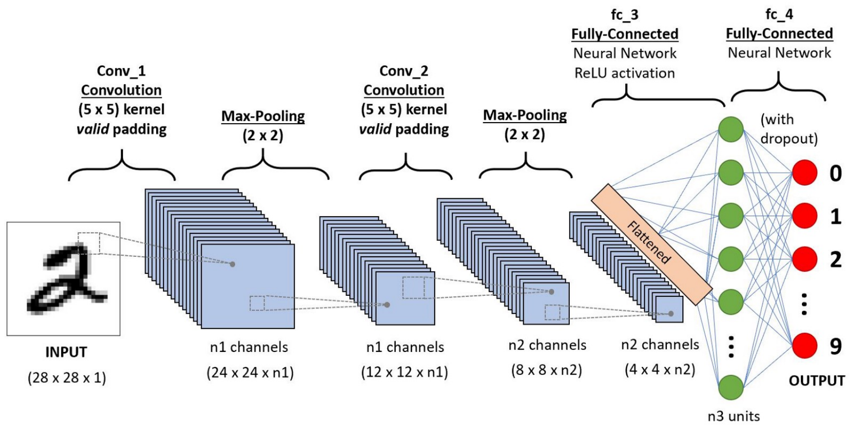


FIGURE 3.1: Convolution Neural Network Architecture.

First layers of a CNN is convolutional layers.In this layer a filter (or kernel) convolve through the input image and create an activation map or feature map.There are two paramenters which comes into play when the filter convolving through the input image.Those are stride and padding.Stride is the number of units that filter should convolve at a time(one step).The padding is used to increase increase the size of input image and keep the resulting output feature map in same size as the input image which will help to preserve more important data for next layer. After convolutional layer data passed through a nonlinear layer(or activation layer).which will bring non linearity to the system.As per the researchers ReLU activation function is more faster and accurate comparing with tanh and sigmoid functions. Mathematically the ReLU function can be represented as y=max(0,X) and the visual representation is illustrated bellow.

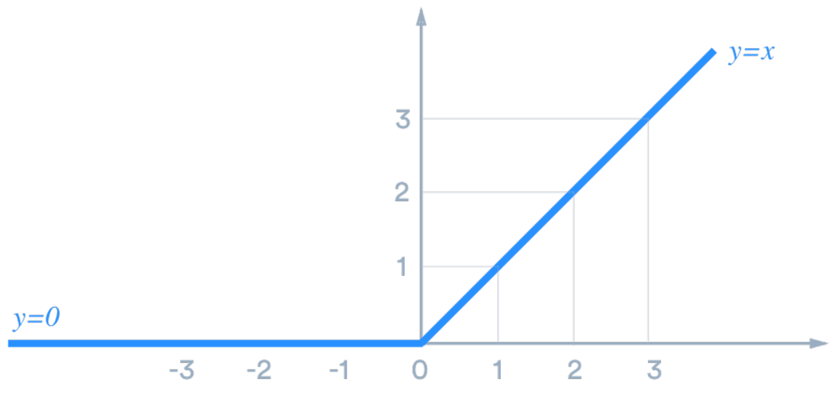


FIGURE 3.2: ReLU activation function.

After the ReLU activation layers data passed through pooling layer and the most popular pooling layer is maxpooling layer where largest value value taken from each segment when filter convole through the data.By using pooling layer the cost of the algorithm could be reduced and it will solve the problem of overfitting.Another layer called dropout layer also used to reduce the overfitting and generalize the algorithm by setting random set of activation to zero. As the last layer of the network fully connected layer works.It takes the output of previous layer and checks the correlation to which class the output more related.With that data an output vector will be created.The length of the vector decided by the number of classes in the classification task.

3.2.1 Convolution Neural Networks performance Improvement

Image classification and object detection is the major requirement to vision based solutions.Huge amount of work going on this subject area.Ouyang used multistage and deformable deep convolution neural network for object detection.Author has used region based convolution neural network in several stages. The approach was to use easy to hard samples as the stages increased.

As per author this architecture improved image classification and object detection[45].Another work done by Cimpoi and team used two convolution neural network based on the classification and object detection problem.The descriptors were FC-CNN and FV-CNN and they have found that orderless pooling of CNN features is quiet good descriptor which could be used in object detection[46].Zhouin his work used technique called class activation mapping(CAM) for CNN with global average pooling CNNs[47].As this many researchers try to improve CNN to use in image classification and object detection.If rich dataset used we could improve the performance of the CNN.To address this it is possible to use a generative model like GAN.

3.3 Generative Adverserial Networks(GANs)

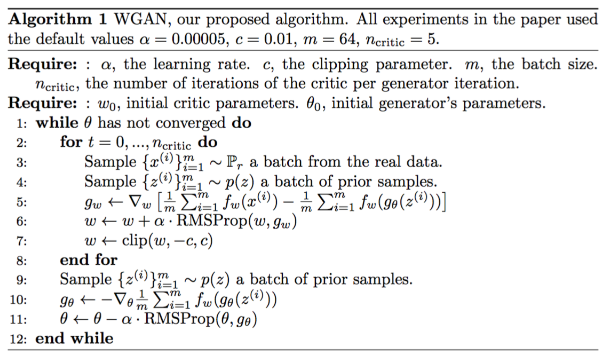
The next important technology used in this music generator is generative adversarial network. GAN contains generative model G and a discriminative model D where generator G tries to fool the discriminator D. In this model generator captures the data distribution while discriminator estimates the probability of a sample came from training data rather than from generator.In here the generator and discriminator plays two player minmax game.In simple term if GAN used for image generation.From a noise vector generator generates a fake image and try to pass it to the discriminator as a real image.Then the discriminator identifies whether the input image is fake or real [48].

3.3.1 Wasserstein Generative Adverserial Networks(WGANs)

Wasserstein GAN is an improved version of the traditional GAN.With this new model the stability of learning is improved as well as algorithm can get rid of mode collapse.Mode collapse is a scenario where the generator generates limited diversity in sample data or generate same data without considering the input data.Wasserstein distance is used to measure the distance between two probability distributions.This measurement also called as Earth Mover′s distance (EM distance).Due to it′s capability to represent smooth distance between two distributions(which were in lower dimensional manifolds without overlaps),it is better than Js or KL divergence.So this EM distance used as the loss function of WGAN. The formula is shown bellow.

$$W(p_r, p_g) = \frac{1}{K} \sup_{\|f\|_L \leq K} \mathbb{E}_{x\sim p_r}[f(x)] - \mathbb{E}_{x\sim p_g}[f(x)] \tag{3.1}$$

Where sup(Supremum) and $\Pi(p_r, p_g)$ is all possible join probability distributions between $p_r$and p_g [49].In this work also WGAN is used to generate new images for the image classification dataset.WGAN pseudo code is illustrated bellow.



Algorithm 1 WGAN, our proposed algorithm. All experiments in the paper used the default values $\alpha = 0.00005$, $c = 0.01$, $m = 64$, $n_{\text{critic}} = 5$.

```
Require: : $\alpha$, the learning rate. $c$, the clipping parameter. $m$, the batch size.
    $n_{\text{critic}}$, the number of iterations of the critic per generator iteration.
Require: : $w_0$, initial critic parameters. $\theta_0$, initial generator's parameters.
 1: while $\theta$ has not converged do
 2:     for $t = 0, ..., n_{\text{critic}}$ do
 3:         Sample $\{x^{(i)}\}_{i=1}^{m} \sim \mathbb{P}_r$ a batch from the real data.
 4:         Sample $\{z^{(i)}\}_{i=1}^{m} \sim p(z)$ a batch of prior samples.
 5:         $g_w \leftarrow \nabla_w \left[\frac{1}{m}\sum_{i=1}^{m} f_w(x^{(i)}) - \frac{1}{m}\sum_{i=1}^{m} f_w(g_\theta(z^{(i)}))\right]$
 6:         $w \leftarrow w + \alpha \cdot \text{RMSProp}(w, g_w)$
 7:         $w \leftarrow \text{clip}(w, -c, c)$
 8:     end for
 9:     Sample $\{z^{(i)}\}_{i=1}^{m} \sim p(z)$ a batch of prior samples.
10:     $g_\theta \leftarrow -\nabla_\theta \frac{1}{m}\sum_{i=1}^{m} f_w(g_\theta(z^{(i)}))$
11:     $\theta \leftarrow \theta - \alpha \cdot \text{RMSProp}(\theta, g_\theta)$
12: end while
```

FIGURE 3.3: WGAN algorithm.

3.3.2 How Generative Adverserial Networks used in image classification

Jonas Natten in his work used GANs to improve the training dataset of face recognition.And the author has achieved 99.42% accuracy which is an improvement of 1.74% compaired to the non improved dataset by GAN[50].Alec Radford and his mates also used deep convolution generative adversarial network for unsupervised learning of image datasets[51]. Dwarikanath Mahapatra and team used conditional GAN for medical image classification task.They used the GAN to generate realistic chest X-ray images[52].Stowell and the team also used GAN to get additional database created for training the network of image classification.According to author acoustic scene classification (ASC) performance using GAN shown impressive performance improvement[53].By going through these works we could use GAN to improve our scene dataset to perform well in scene emotion capturing.

3.4 Long Short Term Memory(LSTM) based music generation

Recurrent neural networks having a problem in preserving memory of last step when stepping in to next step.LSTM can be used as a solution for above mentioned issue where the model can preserve data in a long sequence.Allen Huang and the team used 2 layered LSTM RNN architechture to predict next note of the sequence. And they were able to generate meaning full music[54].Nikhil Kotecha and teamm also used Bi-axial LSTM probabilistic model with neural network to predict and generate polyphonic music[55].So as this researchers have proven that LSTM based solutions could be used in music generation.

3.5 Other libraries and technologies

Since muisc files contain long sequence of data it is better to have a library to handle the basic operations related to musicology. Music21 is such a toolkit for python where we can process sound files easily.Flask is a micro framework which could be used to develop web interface easily.It is extremely flexible framework and contains small core which will reduce cost.Other than these application need more powerful environment to work with.Google cloud platform provide many machine learning related resources and provide easy deployment and maintainability.And also the Google code lab environment could be used in any processing which is done before final deployment.It provide GPU enabled environment to work with.

3.6 Summary

In this chapter I have given detailed information related to the technologies which could be used for this research. That included about convolutional neural networks which could be used with generative adversarial networks to classify scene emotion. And also I have explained how the genetic algorithm could be used in music generation.

Chapter 4

Approach

4.1 Introduction

In chapter 3, detailed introduction about the technologies what could be used in this research is presented . And in this chapter, discuss about the methodology which is followed to achieve the final output of the research. This chapter provides the novel approach to music generation for scene emotion with the use of Generative adversarial technology and LSTM based deep CNN. As such structured subsections namely hypothesis, process, input, output, features of the system presented to explain the approach.

4.2 Hypothesis

Generative adversarial network can be used to improve scene emotion classification. And music can be generated accordingly to the output of classifier using LSTM based CNN. As mentioned earlier chapters music generation completely depends on the output of the image classifier.The requirement of a highly accurate image classifier is high in this project to generate accurate music for the embedded emotion in an input image.According to previous works by researchers that GANs could improve the performance of the classifier it is possible to hypothesize that GANs could be used to improve the performance of the scene emotion classifier.The main drawback of the traditional recurrent neural network is the incapability of preserving important data for future processing of the network(Short memory).So by using LSTM based network it is possible to preserve data in a long sequence.So by using LSTM based model,it is possible to hypothesize that music generation could be done using the available dataset created according to the output of the image classifier.

4.3 Process

The initial process of the system is training the image classifier.To train the image classifier requires a rich dataset which contain high density of quality images for each category(Happy,Sad,Angry).So as a preprocessing task,the Generative adversarial component generates synthetic images and fed those in to the main image dataset which is created using freely available databases in the internet.Then using this database the deep convolutional neural network is trained to classify the images.In the process of classifying images this trained model is used.

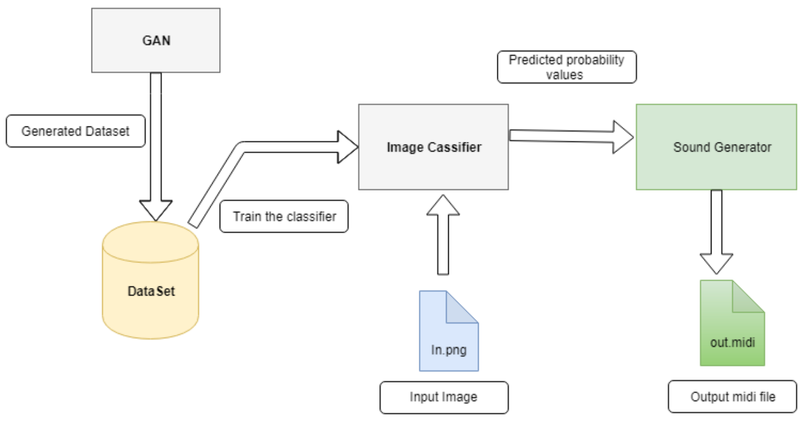


FIGURE 4.1: Proposed solution.

In the main process of the system end user upload an image of a scenery which is randomly selected for music generation as the user preference via the web interface.Using this input image the image classifier classify the image relationships with each category of emotion classes and outputs a vector with the probabilities showing the relatedness for each category. By using this probability vector system extracts sample music dataset for the training and generation process of the music.The music dataset is a midi dataset which is created using songs available in the internet for each category. Then the music generator which is a deep convolutional neural network with LSTM model generates music according to the newly created dataset and outputs a midi sound file.which will be automatically downloaded to the user machine.Due to the high amount of data processing requirement the system is hosted in google cloud environment to achieve high performance.

4.4 Input

Input to the system will be randomly selected image to analyze the Emotion.

4.5 Output

Input image with music clip generated according to the scene emotion as an output which is graphically represented in an application developed.

4.6 Features

Desktop Entertainment Application. User can upload an image and generate music according to the emotion of the image. Fully automated music generation according to scene.

4.7 Users

Administrators who is responsible to train the system with databases and other administrative work on the system. Application end users who are benefited with an entertainment application.

4.8 summary

This chapter described the approach followed to achieve the high performance image classification task and music generation task.And also the details related to users,features of the system,inputs and outputs of the system is presented and described clearly.

Chapter 5

Design

5.1 Introduction

In this chapter explains the high level architecture of the system with details about each component of the project. As explained in chapter 2 the project is developed with 2 phases. First one is the image classifier and the second one is music generator. This chapter will explain all the details about these two component separately.

5.2 Architecture of the project

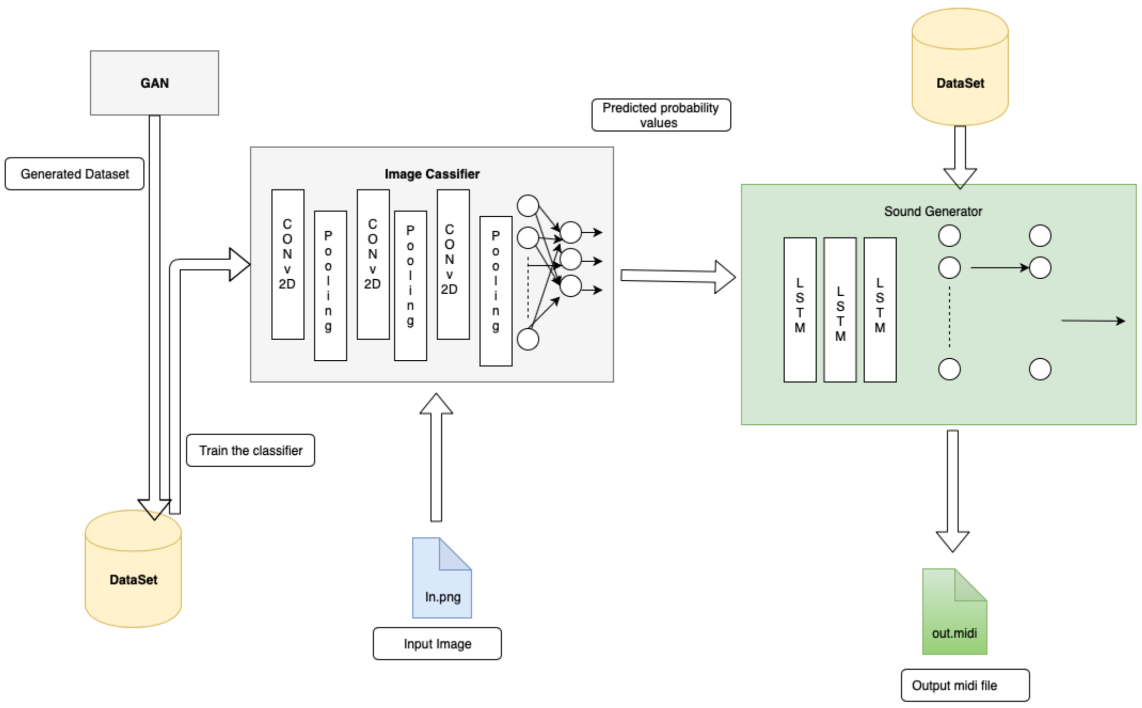


FIGURE 5.1: Architectural Design.

In this design there are two main parts and one supplementary part. The main two parts are image classifier and the music generator and the supplementary part is GAN based image generator.Main data flow has several paths first one is the new scene image generation using GAN.Then image classifier training path.After that comes image prediction and sound generation path. Sound generator also consists training path and generation path.Bellow sequence diagram explains the main flow of the application.

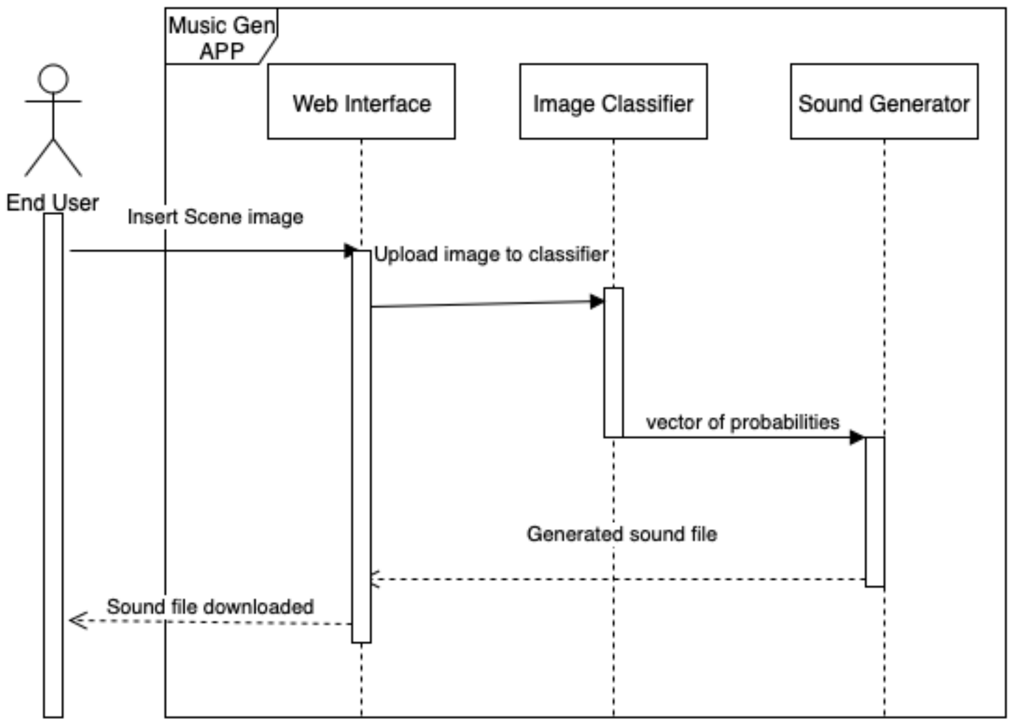


FIGURE 5.2: Sequence Diagram of main flow.

In here user uploads an image using the provided web interface and using the uploaded image image classifier predicts probability values related to each category.Then this output probability vector is passed to music generator and in music generator all the processing parts such as new dataset creation according to probability values,train the system and generating the music takes place after that the final output midi file downloaded to the user machine.

5.3 Image classifier for emotion detection

Image classification model consists of three groups of layers. Those are convolution layers (conv 2D), non linear layer (softmax) and pooling layer (Max pooling 2D).Other than these layers this model consists of two fully connected (Dense) layers. At the first stage we generate supplementary images for our emotions classes and store the output in image database. Then we train the model to classify scene emotions and save the model and using that model we classify the input image.Since we are using softmax activation function the final output will be an array of probabilities for each class.

5.3.1 GAN fake image generator

In here the supplementary part of image classifier is explained which is a WGAN synthetic image generator.Initially the generator starts with noise image and iteration to iteration the image is improved to imitate real image based on the output of the discriminator.In here discriminator always try to identify the input image is a real one or fake one.Basically the training data is related to one category while generating the images. Images generated in this process were fed in to the image classification dataset for classification training task later by the administrator of the application manually.High level architecture of the GAN is illustrated bellow.

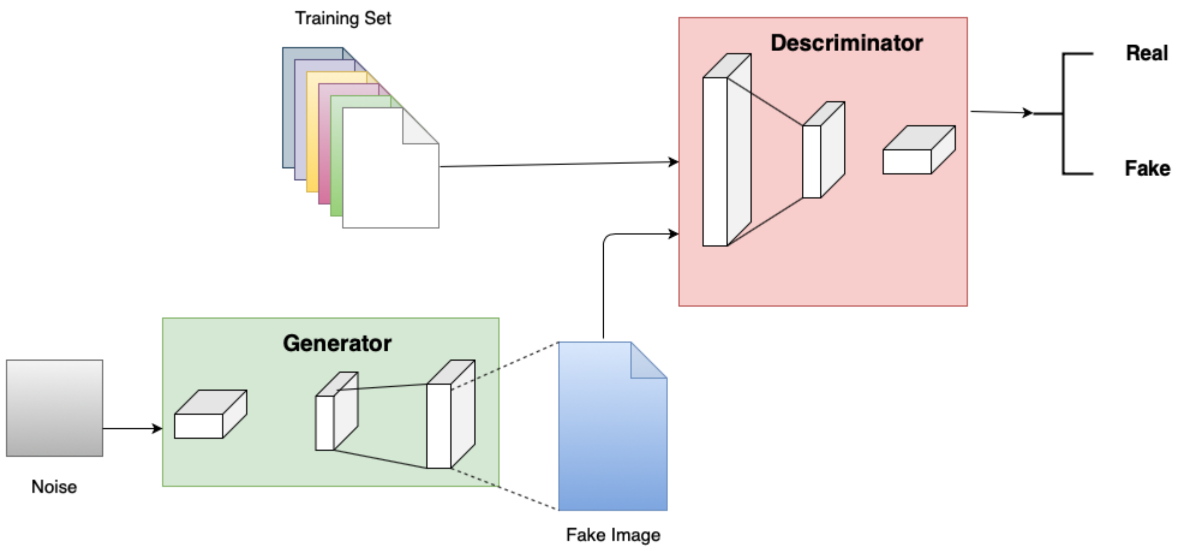


FIGURE 5.3: GAN architecture.

5.4 Music generation

As mentioned above the probability values of image classifier used to decide the dataset which required to use for sound generation.By using probabilities extract dataset randomly from each class and train the LSTM based neural network and using the generated model itis possible to generate music.So the final output is generated after all these processes.

5.5 Summary

In this chapter provided the overall picture of the design of the project. Main technologies used were generative adversarial networks(GANs) and convolutional neural networks(CNNs). In next chapter the implementation information described in detail.

Chapter 6

Implementation

6.1 Introduction

In this chapter the implementation of the music generator for scene emotion is described. Implementation is done using python programming language and used tensorflow, keras libraries to design image classifier.

6.2 Dataset preparation

In this project two dataset were used for image classification and music generation.Image dataset consists scene images which are related to three categories (Sad,Angry,Happy). Combination of several datasets and downloaded images from google used for the project implementation.Used existing datasets were named as UnBiasedEmo and Emotion 6. All the images were color images and we used generative adversarial network to generate more images for all categories as a supplement for the existing dataset. Sound dataset also consist of sound clips for each category in image dataset. All the sound clips extracted from existing songs and converted to 10 seconds midi clips for ease of processing.

6.3 GAN implementation

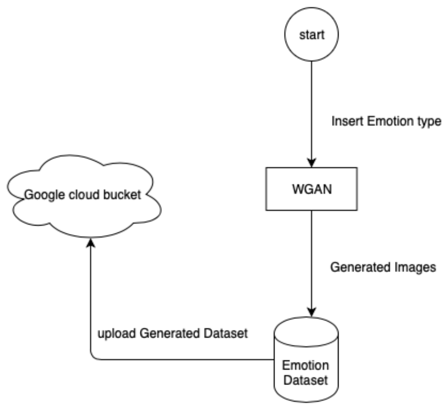


FIGURE 6.1: WGAN image generation flow chart.

As above flow chart initial step of the system is to generate new images for the image classification dataset.When administrator insert the type of the emotion, the system use the dataset related to the emotion and generate new images.Then the images were stored in the database and as the final stage the dataset will be uploaded to the google cloud bucket for the classification task.In here dataset upload is done by administrator manually.Generative adversarial network developed for generation of more images to support each category of the emotions.In here python programming language and several existing libraries as mentioned in introduction used. Generator starting with fake data try to fool the discriminator and using the discriminator output generator learns to generate new images accurately.The code implementation for the generator and discriminator depicted bellow. WGAN implementation used as the image generator for this project specifically.

```
generator(input, random_dim, is_train, reuse=False)
flat_conv1 = tf.add(tf.matmul(input, wx), bx, name='flat_conv1')
conv1 = tf.reshape(flat_conv1, shape=[-1, s4, s4, c4], name='conv1')
bn1 = tf.contrib.layers.batch_norm(conv1, is_training=is_train,
    epsilon=1e-5, decay = 0.9,  updates_collections=None, scope='bn1'
    )
        act1 = tf.nn.relu(bn1, name='act1')
        conv2 = tf.layers.conv2d_transpose(act1, c8, kernel_size=[5,
     5], strides=[2, 2], padding="SAME",
                                           kernel_initializer=tf.
    truncated_normal_initializer(stddev=0.02),
                                           name='conv2')
        bn2 = tf.contrib.layers.batch_norm(conv2, is_training=
    is_train, epsilon=1e-5, decay = 0.9,  updates_collections=None,
    scope='bn2')
        act2 = tf.nn.relu(bn2, name='act2')
        conv3 = tf.layers.conv2d_transpose(act2, c16, kernel_size
    =[5, 5], strides=[2, 2], padding="SAME",
                                           kernel_initializer=tf.
    truncated_normal_initializer(stddev=0.02),
                                           name='conv3')
        bn3 = tf.contrib.layers.batch_norm(conv3, is_training=
    is_train, epsilon=1e-5, decay = 0.9,  updates_collections=None,
    scope='bn3')
        act3 = tf.nn.relu(bn3, name='act3')
        conv4 = tf.layers.conv2d_transpose(act3, c32, kernel_size
    =[5, 5], strides=[2, 2], padding="SAME",
                                           kernel_initializer=tf.
    truncated_normal_initializer(stddev=0.02),
                                           name='conv4')
        bn4 = tf.contrib.layers.batch_norm(conv4, is_training=
    is_train, epsilon=1e-5, decay = 0.9,  updates_collections=None,
    scope='bn4')
        act4 = tf.nn.relu(bn4, name='act4')
        conv5 = tf.layers.conv2d_transpose(act4, c64, kernel_size
    =[5, 5], strides=[2, 2], padding="SAME",
                                           kernel_initializer=tf.
    truncated_normal_initializer(stddev=0.02),
                                           name='conv5')
        bn5 = tf.contrib.layers.batch_norm(conv5, is_training=
    is_train, epsilon=1e-5, decay = 0.9,  updates_collections=None,
    scope='bn5')
        act5 = tf.nn.relu(bn5, name='act5')
        conv6 = tf.layers.conv2d_transpose(act5, output_dim,
    kernel_size=[5, 5], strides=[2, 2], padding="SAME",
                                            kernel_initializer=tf.
    truncated_normal_initializer(stddev=0.02),
                                            name='conv6')
        act6 = tf.nn.tanh(conv6, name='act6')
        return act6

discriminator(input, is_train, reuse=False)
 conv1 = tf.layers.conv2d(input, c2, kernel_size=[5, 5], strides=[2,
     2], padding="SAME",
                                 kernel_initializer=tf.
    truncated_normal_initializer(stddev=0.02),
                                 name='conv1')
        bn1 = tf.contrib.layers.batch_norm(conv1, is_training =
    is_train, epsilon=1e-5, decay = 0.9,  updates_collections=None,
    scope = 'bn1')
        act1 = lrelu(conv1, n='act1')
        conv2 = tf.layers.conv2d(act1, c4, kernel_size=[5, 5],
    strides=[2, 2], padding="SAME",
                                 kernel_initializer=tf.
    truncated_normal_initializer(stddev=0.02),
                                 name='conv2')
        bn2 = tf.contrib.layers.batch_norm(conv2, is_training=
    is_train, epsilon=1e-5, decay = 0.9,  updates_collections=None,
    scope='bn2')
        act2 = lrelu(bn2, n='act2')
        conv3 = tf.layers.conv2d(act2, c8, kernel_size=[5, 5],
    strides=[2, 2], padding="SAME",
                                 kernel_initializer=tf.
    truncated_normal_initializer(stddev=0.02),
                                 name='conv3')
        bn3 = tf.contrib.layers.batch_norm(conv3, is_training=
    is_train, epsilon=1e-5, decay = 0.9,  updates_collections=None,
    scope='bn3')
        act3 = lrelu(bn3, n='act3')
        conv4 = tf.layers.conv2d(act3, c16, kernel_size=[5, 5],
    strides=[2, 2], padding="SAME",
                                 kernel_initializer=tf.
    truncated_normal_initializer(stddev=0.02),
                                 name='conv4')
        bn4 = tf.contrib.layers.batch_norm(conv4, is_training=
    is_train, epsilon=1e-5, decay = 0.9,  updates_collections=None,
    scope='bn4')
        act4 = lrelu(bn4, n='act4')
        dim = int(np.prod(act4.get_shape()[1:]))
        fc1 = tf.reshape(act4, shape=[-1, dim], name='fc1')
        w2 = tf.get_variable('w2', shape=[fc1.shape[-1], 1], dtype=
    tf.float32,
                             initializer=tf.
    truncated_normal_initializer(stddev=0.02))
        b2 = tf.get_variable('b2', shape=[1], dtype=tf.float32,
                                    initializer=tf.constant_initializer
    (0.0))
        logits = tf.add(tf.matmul(fc1, w2), b2, name='logits')
        acted_out = tf.nn.sigmoid(logits)
        return logits
```

6.4 Image classifier implementation

Image classifier also developed using python programming language and using keras library modeled the classifier.The model contains three convolution layers and max pooling layers.The flattened output of above mentioned layers used as input to two fully connected layer.Since we are using softmax activation function the final output of the classifier is an array of probability values.Image classifier model implementation depicted bellow.

```
    model = Sequential()
    model.add(Conv2D(32, (3, 3), input_shape=input_shape))
    model.add(Activation('softmax'))
    model.add(MaxPooling2D(pool_size=(2, 2)))

    model.add(Conv2D(32, (3, 3)))
    model.add(Activation('softmax'))
    model.add(MaxPooling2D(pool_size=(2, 2)))

    model.add(Conv2D(32, (3, 3)))
    model.add(Activation('softmax'))
    model.add(MaxPooling2D(pool_size=(2, 2)))

    model.add(Conv2D(32, (3, 3)))
    model.add(Activation('softmax'))
    model.add(MaxPooling2D(pool_size=(2, 2)))

    model.add(Conv2D(64, (3, 3)))
    model.add(Activation('softmax'))
    model.add(MaxPooling2D(pool_size=(2, 2)))

    model.add(Flatten())
    model.add(Dense(64))
    model.add(Activation('softmax'))
    model.add(Dropout(0.5))
    model.add(Dense(4))
    model.add(Activation('softmax'))
```

6.5 Music generator implementation

Using the output of image classifier system randomly select music clips for each category from music dataset. For the selection process system use probability value of each category.For the ease of handling music related data music 21 library for processing midi clips used in the implementation.Other than that same keras, tensorflow, numpy libraries were used for implementation.In music generator implementation several LSTM layers and 2 fully connected layers used.The model implementation is depicted bellow.

```
 model = Sequential()
 model.add(LSTM(
         256,
         input_shape=(network_input.shape[1], network_input.shape
[2]),
          return_sequences=True
     ))
 model.add(Dropout(0.3))
 model.add(LSTM(512, return_sequences=True))
 model.add(Dropout(0.3))
 model.add(LSTM(256))
 model.add(Dense(256))
 model.add(Dropout(0.3))
 model.add(Dense(n_vocab))
 model.add(Activation('softmax'))
 model.compile(loss='categorical_crossentropy', optimizer='
rmsprop')
```

6.6 Web Api implementation

As the final implementation,created a web api and a web interface to increase the quality of user experience .In web api implementation python and flask library used and hosted it on Google cloud instance to get higher performance and increase accessibility. So the final implementation of main sever file as bellow.

```
app = Flask(__name__)
app.config['SESSION_TYPE'] = 'memcached'
app.config['SECRET_KEY'] = 'super secret key'
app.config['PROJECT_ID'] = 'musicgen'
app.config['CLOUD_STORAGE_BUCKET']='musicgenbkt'

sess = Session()
model = None
image=None

@app.route('/', methods=['GET', 'POST'])
```

```
def home():
    if request.method == 'GET':
        return render_template('home.html')
    if request.method == 'POST':
        if 'image' not in request.files:
            flash('No file was uploaded.')
            return redirect(request.url)
        image_file = request.files['image']
        if image_file.filename == '':
            flash('No file was uploaded.')
            return redirect(request.url)
        qw=trainfiles('trainmidi/Angry/')
        # check if the file is "legit"
        if image_file:
            passed = False
            public_url=''
            try:
                passed = True
                public_url = upload_file(
                    image_file.read(),
                    image_file.filename,
                    image_file.content_type)
            except Exception:
                print("gggg")
                passed = False
            if passed:
                print("call")
                return redirect(url_for('predict', filename=
    image_file.filename), code=307)
            else:
                flash('An error occurred, try again.')
                return redirect(request.url)

@app.route('/predict/<filename>', methods=['POST'])
def predict(filename):
    data = {"success": False}
    print("open")
    url="https://storage.googleapis.com/musicgenbkt/"+filename
    print(url)
    image = imread(url)[:, :, :3]
    image = prepare_image(image, target=(150, 150,3))
    load_modelq()
    with graph.as_default():
        preds = model.predict(image)
        data["predictions"] = []
        a = threading.Thread(target=generate_music(preds), name='
    Thread-a', daemon=True)
        a.start()
    print(preds)
    predictions = preds
 render_template('predict.html',plot_script="",plot_div="",preds=
    predictions,image_url=filename
    )
@app.errorhandler(500)
def server_error(error):
    return render_template('error.html'), 500
```

```
def load_modelq():
    print("load model func called")
    global model
    model = load_model('./models/first_try.model')
    global graph
    graph = tf.get_default_graph()

def prepare_image(image,target):
    image=np.resize(image,target)
    image = img_to_array(image)
    image = image.reshape((1, image.shape[0], image.shape[1], image.
   shape[2]))
    image = preprocess_input(image)
    return image

def trainfiles(prefixa):
    filesarray=[]
    client = get_storage_client()
    bucket = client.bucket(app.config['CLOUD_STORAGE_BUCKET'])
    blobs = bucket.list_blobs(prefix=prefixa, delimiter='/')
    for blob in blobs:
    filesarray.append(blob.name)
    filename = random.choice(filesarray)
    return filesarray

def generate_music(predictions):
    //Music generation implementation
    midi_stream.write('midi', fp='./midigen/test_output.mid')

def get_storage_client():
    return storage.Client(
        project=app.config['PROJECT_ID'])

def upload_file(file_stream, filename, content_type):
  //file upload implementation
    return url
if __name__ == "__main__":
    sess.init_app(app)
    load_modelq()
    app.run(host="127.0.0.1",port=8080,debug=True)
```

6.7 Summary

In this chapter, presented implementation details about the image classifier and music generator in detailed manner. We have used numpy, tensorflow,keras,flask python libraries to support the implementation.

Chapter 7

Evaluation

7.1 Introduction

In this chapter provide all the details about the result what were achieved and how the evaluation procedure what used in this work.And also the outputs of each process.

7.2 Evaluation Procedure

7.2.1 Evaluation Procedure of music files

To evaluate the model a rating system for each sound file generated comparing with scene image used.Sixteen images and sound files generated for them provided and asked the testers to rate the output in 5 categories based on the music quality and relevance to image.The rating categories were excellent,good,fair,poor and bad. Well trained musician and normal listener used to rating process. After they have rated the sound files the mean value of the opinions taken.

7.2.2 Evaluation Procedure of Image classifier

To evaluate the image classifier two sets of image dataset used and calculated the accuracy of using each dataset. The two data sets were raw data set which is having the original images and the other one was dataset which is supplemented by the GAN generated images.

7.3 Image Classification Results

When trying to classify the data with raw dataset system achieved the categorical accuracy of 78% and 80% of validation accuracy after training the network for 1500 iterations.Similar number of iterations were ran to the GAN supplemented data set as well.After creating images for each category to improve learning process. 80% of categorical accuracy and 85% of validation accuracy in image classification task achieved after improving the system with new dataset. Some of the image that were generated as bellow.

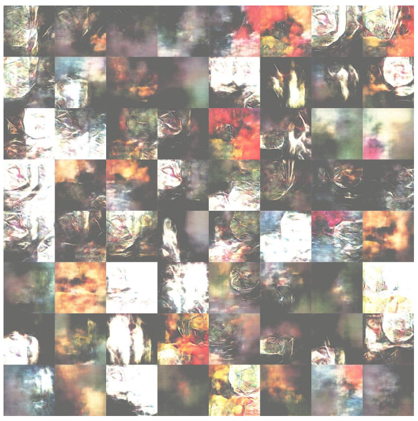


FIGURE 7.1: Generated images for angry.

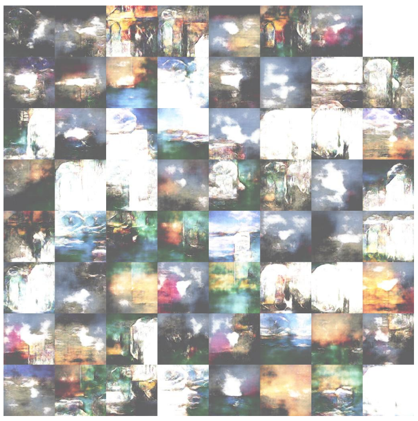


FIGURE 7.2: Generated images for happy.

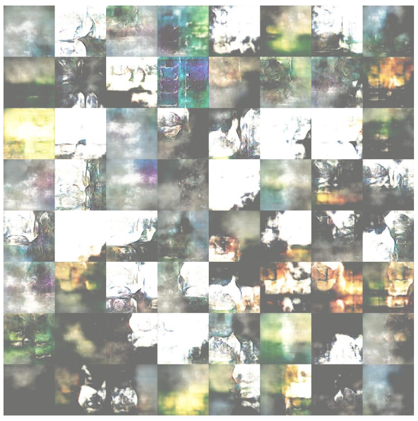


FIGURE 7.3: Generated images for sad.

7.4 Sound Generation Results

Sixteen randomly selected images with sound generated by the system provided to the musicians and normal listener.They have rated each image sound combination with quality values. Since the rating is completely depend on listeners knowledge of music and personal preference this value could get changed.But as a overall idea musician was impressed with generated music and confirmed that generated music is pleasing to hear. The results of each tester is depicted in following table.

TABLE 7.1: Test results of the generated sound files by tester 1.

Test Category	Number of files
Excellent	0
Good	7
Fair	7
Poor	2
Bad	0

TABLE 7.2: Test results of the generated sound files by tester 2.

Test Category	Number of files
Excellent	2
Good	6
Fair	5
Poor	2
Bad	1

TABLE 7.3: Test results of the generated sound files by tester 3.

Test Category	Number of files
Excellent	1
Good	7
Fair	6
Poor	0
Bad	2

By analyzing above results it is visible that more than 50% of the generated sounds categorized by the testers under Excellent , Good and Fair. So these results confirm that the music generation process provide average good results even though the excellent level of sound files were law. And the final overall results were depicted in bellow chart to get more generalized idea using the mean values of the results.

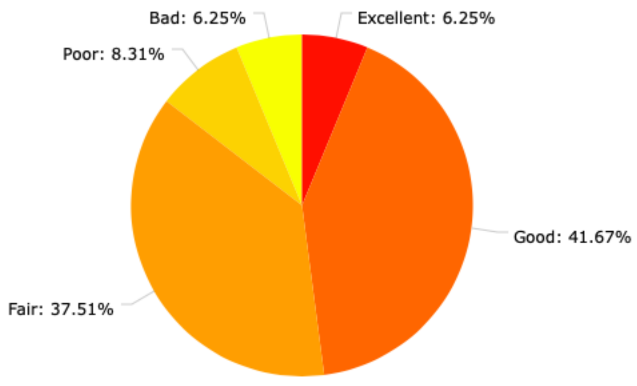


FIGURE 7.4: Results of music evaluation.

As per the mean value results it confirms that the most of the generated data falls under good and fair category.Which also confirms the success of the generated model.

7.5 summary

In this chapter described the test results of generated sound files and provide detailed information about outputs of each part of the system.As per the results the generated sounds were in good quality and the image classifier performance was improved by using supplement dataset created using GAN.

Chapter 8

Conclusion and Future Work

8.1 Introduction

In this chapter provide overall conclusion about the system developed based on evaluated results and suggest future improvements which could be take place.

8.2 Conclusion

In this paper critically studied about the generative models and CNN models and its applicability in the domain of music generation.So generative adversarial networks as a supplement to the existing real image dataset to enhance the classification accuracy used and achieved good results with higher performance.And also critically studied about the existing systems to generate music and acquired relevant knowledge required for the design of the system.

Design of the system done to achieve the performance as well as implementation time constrains.Then the implementation of the system done in two phases. First image classification related implementation and then sound generation related implementation.In here further step taken and created an API to access the implementation.A simple web GUI created to illustrate the results.

Then as next step the system was evaluated with several testers and elaborated the results.Sixteen sound files with the corresponding image provided to the musicians and told them to rank the sound image relationship with the quality of the generated sound.There the system was able achieve 80% of categorical accuracy for the image classifier after using dataset improved by GAN generated images for each category.Since music generated based on the probability values generated by image classifier,the resulting music completely depends on classification capability of classifier. Even though GAN used to improve the dataset,some times the system failed to generate the expected amount of quality in sound file. But as an overall picture system perform well and generated pleasing sounds.As per the testers opinion on generated images more than 50% of the generated sounds were ranked under good and fair.Since the quality check depends on human testers some times the results may bias to some constraints.So even a sound file ranked as bad could be ranked by another tester as fair or good.And as the final stage documented the all the details and produced the final documents.

8.3 Future work

since the music data contain large amount of data it is quiet challenging to process long music clips so if more powerful resources available in time to come the processing time could be reduced and also the length of the clips generated and trained can be increased.So expecting to improve the algorithm and add more processing power and test the model.And also most of the available data is on wave and mp3 formats if it is possible to improve the system to use that kind of raw data it will be more useful.

8.4 summary

In this chapter provided the overall conclusion about the system and suggested future enhancements to the system which are open research areas.

Appendix A

Code Implementations

A.1 Image Classifier Implementation

```python
from keras.preprocessing.image import ImageDataGenerator
from keras.models import Sequential
from keras.layers import Conv2D, MaxPooling2D
from keras.layers import Activation, Dropout, Flatten, Dense
from keras import backend as K


# dimensions of our images.
img_width, img_height = 150, 150

train_data_dir = '/Users/dushanjayawardena/Documents/msc/train'
validation_data_dir = '/Users/dushanjayawardena/Documents/msc/
    validation'
nb_train_samples = 414
nb_validation_samples = 280
epochs = 1000
batch_size = 16

if K.image_data_format() == 'channels_first':
    input_shape = (3, img_width, img_height)
else:
    input_shape = (img_width, img_height, 3)

model = Sequential()
model.add(Conv2D(32, (3, 3), input_shape=input_shape))
model.add(Activation('softmax'))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Conv2D(32, (3, 3)))
model.add(Activation('softmax'))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Conv2D(32, (3, 3)))
model.add(Activation('softmax'))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Conv2D(32, (3, 3)))
model.add(Activation('softmax'))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Conv2D(64, (3, 3)))
model.add(Activation('softmax'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Flatten())
model.add(Dense(64))
model.add(Activation('softmax'))
model.add(Dropout(0.5))
model.add(Dense(3))
model.add(Activation('softmax'))

model.compile(loss='categorical_crossentropy',
              optimizer='rmsprop',
              metrics=['categorical_accuracy'])

# this is the augmentation configuration we will use for training
train_datagen = ImageDataGenerator(
    rescale=1. / 255,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True)

# this is the augmentation configuration we will use for testing:
# only rescaling
test_datagen = ImageDataGenerator(rescale=1. / 255)

train_generator = train_datagen.flow_from_directory(
    train_data_dir,
    target_size=(img_width, img_height),
    batch_size=batch_size,
    class_mode='categorical')

validation_generator = test_datagen.flow_from_directory(
    validation_data_dir,
    target_size=(img_width, img_height),
    batch_size=batch_size,
    class_mode='categorical')

model.fit_generator(
    train_generator,
    steps_per_epoch=nb_train_samples // batch_size,
    epochs=epochs,
    validation_data=validation_generator,
    validation_steps=nb_validation_samples // batch_size)

model.save_weights('/Users/dushanjayawardena/Documents/first_try.h5'
    )
model.save('/Users/dushanjayawardena/Documents/first_try.model')
```

A.2 Music Generator

```
from music21 import converter, instrument, note, chord,stream
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import Dropout
from keras.layers import LSTM
from keras.layers import Activation
from keras.utils import np_utils
from keras.callbacks import ModelCheckpoint
from keras.models import load_model
from keras.preprocessing.image import load_img
from keras.preprocessing.image import img_to_array
from keras.applications.vgg16 import preprocess_input
from keras.applications import imagenet_utils
import tensorflow as tf
from PIL import Image
import numpy as np
import flask
import os, random
import io
import glob
import pickle


# initialize our Flask application and the Keras model
app = flask.Flask(__name__)
model = None

def load_modelq():
  # load the pre-trained Keras model (here we are using a model
  # pre-trained on ImageNet and provided by Keras, but you can
  # substitute in your own networks just as easily)
    global model
    model = load_model('/Users/dushanjayawardena/Documents/first_try
   .model')
    global graph
    graph = tf.get_default_graph()

def prepare_image(image,target):
  # if the image mode is not RGB, convert it
  if image.mode != "RGB":
    image = image.convert("RGB")

  # resize the input image and preprocess it
  image = image.resize(target)
  image = img_to_array(image)
  image = image.reshape((1, image.shape[0], image.shape[1], image.
   shape[2]))
  image = preprocess_input(image)

  # return the processed image
  return image
def generate_music(predictions):
    notes = []
    print('start of file read');
    for x in range(int(predictions[0][0]*1000)):
    #for file in glob.glob("/content/driver1/My Drive/msc/trainmidi/
   trainmidi/*.mid"):
        #print(file);
        filename = random.choice(os.listdir("/Users/
   dushanjayawardena/Documents/msc/Angry"))
```

```
    print(filename);
    file = os.path.join("/Users/dushanjayawardena/Documents/msc/Angry", filename)
    midi = converter.parse(file)
    notes_to_parse = None
    parts = instrument.partitionByInstrument(midi)
    if parts: # file has instrument parts
        notes_to_parse = parts.parts[0].recurse()
    else: # file has notes in a flat structure
        notes_to_parse = midi.flat.notes
    for element in notes_to_parse:
        if isinstance(element, note.Note):
            notes.append(str(element.pitch))
        elif isinstance(element, chord.Chord):
            notes.append('.'.join(str(n) for n in element.normalOrder))
################################
for x in range(int(predictions[0][1]*1000)):
#for file in glob.glob("/content/driver1/My Drive/msc/trainmidi/trainmidi/*.mid"):
    #print(file);
    filename = random.choice(os.listdir("/Users/dushanjayawardena/Documents/msc/Happy"))
    print(filename);
    file = os.path.join("/Users/dushanjayawardena/Documents/msc/Happy", filename)
    midi = converter.parse(file)
    notes_to_parse = None
    parts = instrument.partitionByInstrument(midi)
    if parts: # file has instrument parts
        notes_to_parse = parts.parts[0].recurse()
    else: # file has notes in a flat structure
        notes_to_parse = midi.flat.notes
    for element in notes_to_parse:
        if isinstance(element, note.Note):
            notes.append(str(element.pitch))
        elif isinstance(element, chord.Chord):
            notes.append('.'.join(str(n) for n in element.normalOrder))
######################################
for x in range(int(predictions[0][2]*1000)):
#for file in glob.glob("/content/driver1/My Drive/msc/trainmidi/trainmidi/*.mid"):
    #print(file);
    filename = random.choice(os.listdir("/Users/dushanjayawardena/Documents/msc/Happy"))
    file = os.path.join("/Users/dushanjayawardena/Documents/msc/Happy", filename)
    midi = converter.parse(file)
    notes_to_parse = None
    parts = instrument.partitionByInstrument(midi)
    if parts: # file has instrument parts
        notes_to_parse = parts.parts[0].recurse()
    else: # file has notes in a flat structure
        notes_to_parse = midi.flat.notes
    for element in notes_to_parse:
```

```
             if isinstance(element, note.Note):
                 notes.append(str(element.pitch))
             elif isinstance(element, chord.Chord):
                 notes.append('.'.join(str(n) for n in element.
normalOrder))
 ##########################################
 for x in range(int(predictions[0][3]*1000)):
 #for file in glob.glob("/content/driver1/My Drive/msc/trainmidi/
trainmidi/*.mid"):
     #print(file);
     filename = random.choice(os.listdir("/Users/
dushanjayawardena/Documents/msc/Sad"))
     print(filename);
     file = os.path.join("/Users/dushanjayawardena/Documents/msc/
Sad", filename)
     midi = converter.parse(file)
     notes_to_parse = None
     parts = instrument.partitionByInstrument(midi)
     if parts: # file has instrument parts
         notes_to_parse = parts.parts[0].recurse()
     else: # file has notes in a flat structure
         notes_to_parse = midi.flat.notes
     for element in notes_to_parse:
         if isinstance(element, note.Note):
             notes.append(str(element.pitch))
         elif isinstance(element, chord.Chord):
             notes.append('.'.join(str(n) for n in element.
normalOrder))
 print('end of file read');
 sequence_length = 100
 n_vocab = len(set(notes))
 #print(n_vocab);
 # get all pitch names
 pitchnames = sorted(set(item for item in notes))
 # create a dictionary to map pitches to integers
 note_to_int = dict((note, number) for number, note in enumerate(
pitchnames))
 network_input = []
 network_output = []
 # create input sequences and the corresponding outputs
 for i in range(0, len(notes) - sequence_length, 1):
     sequence_in = notes[i:i + sequence_length]
     sequence_out = notes[i + sequence_length]
     network_input.append([note_to_int[char] for char in
sequence_in])
     network_output.append(note_to_int[sequence_out])
     #print(len(network_output))
 n_patterns = len(network_input)
 # reshape the input into a format compatible with LSTM layers
 network_input = np.reshape(network_input, (n_patterns,
sequence_length, 1))
 # normalize input
 network_input = network_input / float(n_vocab)
 print(len(network_output))
 network_output = np_utils.to_categorical(network_output)

```

```
model = Sequential()
model.add(LSTM(
        256,
        input_shape=(network_input.shape[1], network_input.shape[2]),
        return_sequences=True
    ))
model.add(Dropout(0.3))
model.add(LSTM(512, return_sequences=True))
model.add(Dropout(0.3))
model.add(LSTM(256))
model.add(Dense(256))
model.add(Dropout(0.3))
model.add(Dense(n_vocab))
model.add(Activation('softmax'))
model.compile(loss='categorical_crossentropy', optimizer='rmsprop')

filepath = "/Users/dushanjayawardena/Documents/msc/trainmodel/weights-improvement-{epoch:02d}-{loss:.4f}-bigger.hdf5"
checkpoint = ModelCheckpoint(
    filepath, monitor='loss',
    verbose=0,
    save_best_only=True,
    mode='min'
)
callbacks_list = [checkpoint]
model.fit(network_input, network_output, epochs=20, batch_size=64, callbacks=callbacks_list)
model.save_weights('/Users/dushanjayawardena/Documents/msc/trainmodel/weights.hdf5')
model.load_weights('/Users/dushanjayawardena/Documents/msc/trainmodel/weights.hdf5')

start = np.random.randint(0, len(network_input)-1)
int_to_note = dict((number, note) for number, note in enumerate(pitchnames))
pattern = network_input[start]
prediction_output = []
# generate 500 notes
for note_index in range(500):
    prediction_input = np.reshape(pattern, (1, len(pattern), 1))
    prediction_input = prediction_input / float(n_vocab)
    prediction = model.predict(prediction_input, verbose=0)
    index = np.argmax(prediction)
    result = int_to_note[index]
    prediction_output.append(result)
    pattern=np.append(pattern,index)
    #pattern.append(index)
    pattern = pattern[1:len(pattern)]

offset = 0
output_notes = []
# create note and chord objects based on the values generated by the model
```

```
    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)
        # increase offset each iteration so that notes do not stack
        offset += 0.5

    midi_stream = stream.Stream(output_notes)
    midi_stream.write('midi', fp='/Users/dushanjayawardena/Documents
   /msc/test_output.mid')

@app.route("/predict", methods=["POST"])
def predict():
    # initialize the data dictionary that will be returned from the
    # view
    data = {"success": False}

    # ensure an image was properly uploaded to our endpoint
    if flask.request.method == "POST":
        if flask.request.files.get("image"):
            # read the image in PIL format
            image = flask.request.files["image"].read()
            image = Image.open(io.BytesIO(image))

      # preprocess the image and prepare it for classification
            image = prepare_image(image, target=(150, 150))
            with graph.as_default():
      # classify the input image and then initialize the list
      # of predictions to return to the client
                preds = model.predict(image)

                data["predictions"] = []

      # loop over the results and add them to the list of
      # returned predictions

                r = {"probability": int(preds[0][0])}
                data["predictions"].append(r)
                r = {"probability": float(preds[0][1])}
                data["predictions"].append(r)
                r = {"probability": float(preds[0][2])}
                data["predictions"].append(r)
                r = {"probability": float(preds[0][3])}
                data["predictions"].append(r)

        # indicate that the request was a success
                data["success"] = True
                generate_music(preds)
    # return the data dictionary as a JSON response
    return flask.jsonify(data)


# if this is the main thread of execution first load the model and
# then start the server
if __name__ == "__main__":
    print(("* Loading Keras model and Flask starting server..."
        "please wait until server has fully started"))
    load_modelq()
    app.run()
```

A.3 GAN Implementation

```

def generator(input, random_dim, is_train, reuse=False):
    c4, c8, c16, c32, c64 = 512, 256, 128, 64, 32 # channel num
    s4 = 4
    output_dim = CHANNEL  # RGB image
    with tf.variable_scope('gen') as scope:
        if reuse:
            scope.reuse_variables()
        wx = tf.get_variable('wx', shape=[random_dim, s4 * s4 * c4],
     dtype=tf.float32,
                                 initializer=tf.
    truncated_normal_initializer(stddev=0.02))
        bx = tf.get_variable('bx', shape=[c4 * s4 * s4], dtype=tf.
    float32,
                                 initializer=tf.constant_initializer
    (0.0))
        flat_conv1 = tf.add(tf.matmul(input, wx), bx, name='
    flat_conv1')
         #Convolution, bias, activation, repeat!
        conv1 = tf.reshape(flat_conv1, shape=[-1, s4, s4, c4], name=
    'conv1')
        bn1 = tf.contrib.layers.batch_norm(conv1, is_training=
    is_train, epsilon=1e-5, decay = 0.9,  updates_collections=None,
    scope='bn1')
        act1 = tf.nn.relu(bn1, name='act1')
        # 8*8*256
        #Convolution, bias, activation, repeat!
        conv2 = tf.layers.conv2d_transpose(act1, c8, kernel_size=[5,
     5], strides=[2, 2], padding="SAME",
                                           kernel_initializer=tf.
    truncated_normal_initializer(stddev=0.02),
                                                    name='conv2')
        bn2 = tf.contrib.layers.batch_norm(conv2, is_training=
    is_train, epsilon=1e-5, decay = 0.9,  updates_collections=None,
    scope='bn2')
        act2 = tf.nn.relu(bn2, name='act2')
        # 16*16*128
        conv3 = tf.layers.conv2d_transpose(act2, c16, kernel_size
    =[5, 5], strides=[2, 2], padding="SAME",
                                                    kernel_initializer=tf.
    truncated_normal_initializer(stddev=0.02),
                                                    name='conv3')
        bn3 = tf.contrib.layers.batch_norm(conv3, is_training=
    is_train, epsilon=1e-5, decay = 0.9,  updates_collections=None,
    scope='bn3')
        act3 = tf.nn.relu(bn3, name='act3')
        # 32*32*64
        conv4 = tf.layers.conv2d_transpose(act3, c32, kernel_size
    =[5, 5], strides=[2, 2], padding="SAME",
                                                    kernel_initializer=tf.
    truncated_normal_initializer(stddev=0.02),
                                                    name='conv4')
        bn4 = tf.contrib.layers.batch_norm(conv4, is_training=
    is_train, epsilon=1e-5, decay = 0.9,  updates_collections=None,
    scope='bn4')
        act4 = tf.nn.relu(bn4, name='act4')
        # 64*64*32
        conv5 = tf.layers.conv2d_transpose(act4, c64, kernel_size
    =[5, 5], strides=[2, 2], padding="SAME",
                                                    kernel_initializer=tf.
    truncated_normal_initializer(stddev=0.02),
                                                    name='conv5')
        bn5 = tf.contrib.layers.batch_norm(conv5, is_training=
    is_train, epsilon=1e-5, decay = 0.9,  updates_collections=None,
    scope='bn5')
        act5 = tf.nn.relu(bn5, name='act5')

        #128*128*3
        conv6 = tf.layers.conv2d_transpose(act5, output_dim,
    kernel_size=[5, 5], strides=[2, 2], padding="SAME",
                                                    kernel_initializer=tf.
    truncated_normal_initializer(stddev=0.02),
                                                    name='conv6')
        # bn6 = tf.contrib.layers.batch_norm(conv6, is_training=
    is_train, epsilon=1e-5, decay = 0.9,  updates_collections=None,
    scope='bn6')
        act6 = tf.nn.tanh(conv6, name='act6')
        return act6


def discriminator(input, is_train, reuse=False):
    c2, c4, c8, c16 = 64, 128, 256, 512  # channel num: 64, 128,
    256, 512
    with tf.variable_scope('dis') as scope:
        if reuse:
            scope.reuse_variables()
        #Convolution, activation, bias, repeat!
        conv1 = tf.layers.conv2d(input, c2, kernel_size=[5, 5],
strides=[2, 2], padding="SAME",
                                         kernel_initializer=tf.
truncated_normal_initializer(stddev=0.02),
                                         name='conv1')
        bn1 = tf.contrib.layers.batch_norm(conv1, is_training =
is_train, epsilon=1e-5, decay = 0.9,  updates_collections=None,
scope = 'bn1')
        act1 = lrelu(conv1, n='act1')
         #Convolution, activation, bias, repeat!
        conv2 = tf.layers.conv2d(act1, c4, kernel_size=[5, 5],
strides=[2, 2], padding="SAME",
                                         kernel_initializer=tf.
truncated_normal_initializer(stddev=0.02),
                                         name='conv2')
        bn2 = tf.contrib.layers.batch_norm(conv2, is_training=
is_train, epsilon=1e-5, decay = 0.9,  updates_collections=None,
scope='bn2')
        act2 = lrelu(bn2, n='act2')
        #Convolution, activation, bias, repeat!
        conv3 = tf.layers.conv2d(act2, c8, kernel_size=[5, 5],
strides=[2, 2], padding="SAME",
                                         kernel_initializer=tf.
truncated_normal_initializer(stddev=0.02),
                                         name='conv3')
        bn3 = tf.contrib.layers.batch_norm(conv3, is_training=
is_train, epsilon=1e-5, decay = 0.9,  updates_collections=None,
scope='bn3')
        act3 = lrelu(bn3, n='act3')
         #Convolution, activation, bias, repeat!
        conv4 = tf.layers.conv2d(act3, c16, kernel_size=[5, 5],
strides=[2, 2], padding="SAME",
                                         kernel_initializer=tf.
truncated_normal_initializer(stddev=0.02),
                                         name='conv4')
        bn4 = tf.contrib.layers.batch_norm(conv4, is_training=
is_train, epsilon=1e-5, decay = 0.9,  updates_collections=None,
scope='bn4')
        act4 = lrelu(bn4, n='act4')

        # start from act4
        dim = int(np.prod(act4.get_shape()[1:]))
        fc1 = tf.reshape(act4, shape=[-1, dim], name='fc1')


        w2 = tf.get_variable('w2', shape=[fc1.shape[-1], 1], dtype=
tf.float32,
                                  initializer=tf.
truncated_normal_initializer(stddev=0.02))
        b2 = tf.get_variable('b2', shape=[1], dtype=tf.float32,
                                  initializer=tf.constant_initializer
(0.0))

        # wgan just get rid of the sigmoid
        logits = tf.add(tf.matmul(fc1, w2), b2, name='logits')
        # dcgan
        acted_out = tf.nn.sigmoid(logits)
        return logits #, acted_out
```

Bibliography

[1] IAN CROSS. “Music, Cognition, Culture, and Evolution”. In: (), p. 15.

[2] Dragan Matic. “A genetic algorithm for composing music”. In: *Yugoslav Journal of Operations Research* 20.1 (2010), pp. 157–177. ISSN: 0354-0243. DOI: 10.2298/YJOR1001157M. URL: http://www.doiserbia.nb.rs/Article.aspx?ID=0354-02431001157M (visited on 04/07/2018).

[3] Jônatas Manzolli et al. “A METHOD FOR SOUND SYNTHESIS BASED ON GENETIC ALGORITHMS”. In: (), p. 10.

[4] Xiaoying Wu and Ze-Nian Li. “2-A study of image-based music composition”. In: IEEE, June 2008, pp. 1345–1348. ISBN: 978-1-4244-2570-9. DOI: 10.1109/ICME.2008.4607692. URL: http://ieeexplore.ieee.org/document/4607692/ (visited on 04/07/2018).

[5] Koen E. A. van de Sande, Theo Gevers, and Cees G. M. Snoek. *Evaluating Color Descriptors for Object and Scene Recognition*. 2009.

[6] Szabolcs Sergyán. *3-Color Content-based Image Classification*.

[7] Hye-Rin Kim et al. “1-Building Emotional Machines: Recognizing Image Emotions through Deep Neural Networks”. In: *arXiv:1705.07543 [cs]* (May 21, 2017). arXiv: 1705.07543. URL: http://arxiv.org/abs/1705.07543 (visited on 05/14/2018).

[8] Bolei Zhou et al. “5-Learning Deep Features for Scene Recognition using Places Database”. In: (), p. 9.

[9] Xinyue Zhu et al. “4-Data Augmentation in Emotion Classification Using Generative Adversarial Networks”. In: *arXiv:1711.00648 [cs]* (Nov. 2, 2017). arXiv: 1711.00648. URL: http://arxiv.org/abs/1711.00648 (visited on 04/03/2018).

[10] Junyoung Chung et al. “Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling”. In: *arXiv:1412.3555 [cs]* (Dec. 11, 2014). arXiv: 1412.3555. URL: http://arxiv.org/abs/1412.3555 (visited on 05/26/2018).

[11] Alex Graves. “Generating Sequences With Recurrent Neural Networks”. In: *arXiv:1308.0850 [cs]* (Aug. 4, 2013). arXiv: 1308.0850. URL: http://arxiv.org/abs/1308.0850 (visited on 05/26/2018).

[12] Nicolas Boulanger-Lewandowski, Yoshua Bengio, and Pascal Vincent. “Modeling Temporal Dependencies in High-Dimensional Sequences: Application to Polyphonic Music Generation and Transcription”. In: (), p. 8.

[13] Natasha Jaques et al. “Sequence Tutor: Conservative Fine-Tuning of Sequence Generation Models with KL-control”. In: *arXiv:1611.02796 [cs]* (Nov. 8, 2016). arXiv: 1611.02796. URL: http://arxiv.org/abs/1611.02796 (visited on 05/26/2018).

[14] Adhika Sigit Ramanto and Jl Ganesha No. “Markov Chain Based Procedural Music Generator with User Chosen Mood Compatibility”. In: (), p. 6.

[15] S.-B. Cho. “6-Emotional Image and Musical Information Retrieval With Interactive Genetic Algorithm”. In: *Proceedings of the IEEE* 92.4 (Apr. 2004),

pp. 702–711. ISSN: 0018-9219. DOI: 10.1109/JPROC.2004.825900. URL: http://ieeexplore.ieee.org/document/1278692/ (visited on 05/19/2018).

[16] Saber Malekzadeh et al. "Classical Music Generation in Distinct Dastgahs with AlimNet ACGAN". In: (), p. 5.

[17] Sander Dieleman, Aäron van den Oord, and Karen Simonyan. "The challenge of realistic music generation: modelling raw audio at scale". In: *arXiv:1806.10474 [cs, eess, stat]* (June 26, 2018). arXiv: 1806.10474. URL: http://arxiv.org/abs/1806.10474 (visited on 01/25/2019).

[18] Kevin Joslyn, Naifan Zhuang, and Kien A. Hua. "Deep Segment Hash Learning for Music Generation". In: *arXiv:1805.12176 [cs, stat]* (May 30, 2018). arXiv: 1805.12176. URL: http://arxiv.org/abs/1805.12176 (visited on 01/25/2019).

[19] Stefan Lattner, Maarten Grachten, and Gerhard Widmer. "Imposing higher-level Structure in Polyphonic Music Generation using Convolutional Restricted Boltzmann Machines and Constraints". In: *Journal of Creative Music Systems* 2.1 (Mar. 31, 2018). ISSN: 23997656. DOI: 10.5920/jcms.2018.01. arXiv: 1612.04742. URL: http://arxiv.org/abs/1612.04742 (visited on 01/25/2019).

[20] Jay A. Hennig, Akash Umakantha, and Ryan C. Williamson. "A Classifying Variational Autoencoder with Application to Polyphonic Music Generation". In: *arXiv:1711.07050 [cs, stat]* (Nov. 19, 2017). arXiv: 1711.07050. URL: http://arxiv.org/abs/1711.07050 (visited on 01/25/2019).

[21] Kratarth Goel, Raunaq Vohra, and J. K. Sahoo. "Polyphonic Music Generation by Modeling Temporal Dependencies Using a RNN-DBN". In: *arXiv:1412.7927 [cs]* 8681 (2014), pp. 217–224. DOI: 10.1007/978-3-319-11179-7_28. arXiv: 1412.7927. URL: http://arxiv.org/abs/1412.7927 (visited on 01/25/2019).

[22] Mason Bretan, Gil Weinberg, and Larry Heck. "A Unit Selection Methodology for Music Generation Using Deep Neural Networks". In: *arXiv:1612.03789 [cs]* (Dec. 2016). arXiv: 1612.03789. URL: http://arxiv.org/abs/1612.03789 (visited on 03/22/2019).

[23] Florian Colombo, Alexander Seeholzer, and Wulfram Gerstner. "Deep Artificial Composer: A Creative Neural Network Model for Automated Melody Generation". en. In: *Computational Intelligence in Music, Sound, Art and Design*. Ed. by João Correia, Vic Ciesielski, and Antonios Liapis. Vol. 10198. Cham: Springer International Publishing, 2017, pp. 81–96. ISBN: 978-3-319-55749-6 978-3-319-55750-2. DOI: 10.1007/978-3-319-55750-2_6. URL: http://link.springer.com/10.1007/978-3-319-55750-2_6 (visited on 03/22/2019).

[24] Huanru Henry Mao, Taylor Shin, and Garrison W. Cottrell. "DeepJ: Style-Specific Music Generation". In: *2018 IEEE 12th International Conference on Semantic Computing (ICSC)* (Jan. 2018). arXiv: 1801.00887, pp. 377–382. DOI: 10.1109/ICSC.2018.00077. URL: http://arxiv.org/abs/1801.00887 (visited on 03/22/2019).

[25] Gino Brunner et al. "JamBot: Music Theory Aware Chord Based Generation of Polyphonic Music with LSTMs". In: *arXiv:1711.07682 [cs, eess,*

math, stat] (Nov. 2017). arXiv: 1711.07682. URL: http://arxiv.org/abs/1711.07682 (visited on 03/22/2019).

[26] Vasanth Kalingeri and Srikanth Grandhe. "Music Generation with Deep Learning". In: *arXiv:1612.04928 [cs]* (Dec. 2016). arXiv: 1612.04928. URL: http://arxiv.org/abs/1612.04928 (visited on 03/22/2019).

[27] Qi Lyu, Zhiyong Wu, and Jun Zhu. "Polyphonic Music Modelling with LSTM-RTRBM". en. In: *Proceedings of the 23rd ACM international conference on Multimedia - MM '15*. Brisbane, Australia: ACM Press, 2015, pp. 991–994. ISBN: 978-1-4503-3459-4. DOI: 10.1145/2733373.2806383. URL: http://dl.acm.org/citation.cfm?doid=2733373.2806383 (visited on 03/22/2019).

[28] Hang Chu, Raquel Urtasun, and Sanja Fidler. "Song From PI: A Musically Plausible Network for Pop Music Generation". In: *arXiv:1611.03477 [cs]* (Nov. 2016). arXiv: 1611.03477. URL: http://arxiv.org/abs/1611.03477 (visited on 03/22/2019).

[29] Jana Machajdik and Allan Hanbury. "5-Affective image classification using features inspired by psychology and art theory". In: ACM Press, 2010, p. 83. ISBN: 978-1-60558-933-6. DOI: 10.1145/1873951.1873965. URL: http://dl.acm.org/citation.cfm?doid=1873951.1873965 (visited on 05/19/2018).

[30] Xin Lu et al. "On shape and the computability of emotions". In: ACM Press, 2012, p. 229. ISBN: 978-1-4503-1089-5. DOI: 10.1145/2393347.2393384. URL: http://dl.acm.org/citation.cfm?doid=2393347.2393384 (visited on 05/26/2018).

[31] Tao Chen et al. "DeepSentiBank: Visual Sentiment Concept Classification with Deep Convolutional Neural Networks". In: *arXiv:1410.8586 [cs]* (Oct. 30, 2014). arXiv: 1410.8586. URL: http://arxiv.org/abs/1410.8586 (visited on 05/26/2018).

[32] Yongli Zhu, Chengxi Liu, and Kai Sun. "Image Embedding of PMU Data for Deep Learning Towards Transient Disturbance Classification". In: *2018 IEEE International Conference on Energy Internet (ICEI)*. 2018 2nd IEEE International Conference on Energy Internet (ICEI). Beijing: IEEE, May 2018, pp. 169–174. ISBN: 978-1-5386-4131-6. DOI: 10.1109/ICEI.2018.00038. URL: https://ieeexplore.ieee.org/document/8403446/ (visited on 01/25/2019).

[33] Arindam Das et al. "Document Image Classification with Intra-Domain Transfer Learning and Stacked Generalization of Deep Convolutional Neural Networks". In: *arXiv:1801.09321 [cs]* (Jan. 28, 2018). arXiv: 1801.09321. URL: http://arxiv.org/abs/1801.09321 (visited on 01/25/2019).

[34] Roshan Gopalakrishnan, Yansong Chua, and Laxmi R. Iyer. "Classifying neuromorphic data using a deep learning framework for image classification". In: *arXiv:1807.00578 [cs]* (July 2, 2018). arXiv: 1807.00578. URL: http://arxiv.org/abs/1807.00578 (visited on 01/25/2019).

[35] Alhussein Fawzi et al. "Adaptive data augmentation for image classification". en. In: *2016 IEEE International Conference on Image Processing (ICIP)*.

Phoenix, AZ, USA: IEEE, Sept. 2016, pp. 3688–3692. ISBN: 978-1-4673-9961-6. DOI: 10.1109/ICIP.2016.7533048. URL: http://ieeexplore.ieee.org/document/7533048/ (visited on 03/22/2019).

[36] Ren Wu et al. "Deep Image: Scaling up Image Recognition". en. In: (), p. 12.

[37] Saining Xie et al. "Hyper-class augmented and regularized deep learning for fine-grained image classification". en. In: *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Boston, MA, USA: IEEE, June 2015, pp. 2645–2654. ISBN: 978-1-4673-6964-0. DOI: 10.1109/CVPR.2015.7298880. URL: http://ieeexplore.ieee.org/document/7298880/ (visited on 03/22/2019).

[38] Luke Taylor and Geoff Nitschke. "Improving Deep Learning with Generic Data Augmentation". en. In: *2018 IEEE Symposium Series on Computational Intelligence (SSCI)*. Bangalore, India: IEEE, Nov. 2018, pp. 1542–1547. ISBN: 978-1-5386-9276-9. DOI: 10.1109/SSCI.2018.8628742. URL: https://ieeexplore.ieee.org/document/8628742/ (visited on 03/22/2019).

[39] Zhun Zhong et al. "Random Erasing Data Augmentation". In: *arXiv:1708.04896 [cs]* (Aug. 2017). arXiv: 1708.04896. URL: http://arxiv.org/abs/1708.04896 (visited on 03/22/2019).

[40] Sebastien C. Wong et al. "Understanding data augmentation for classification: when to warp?" en. In: *arXiv:1609.08764 [cs]* (Sept. 2016). arXiv: 1609.08764. URL: http://arxiv.org/abs/1609.08764 (visited on 03/22/2019).

[41] Antreas Antoniou, Amos Storkey, and Harrison Edwards. "Data Augmentation Generative Adversarial Networks". In: *arXiv:1711.04340 [cs, stat]* (Nov. 2017). arXiv: 1711.04340. URL: http://arxiv.org/abs/1711.04340 (visited on 03/22/2019).

[42] Mina Rezaei et al. "Conditional Adversarial Network for Semantic Segmentation of Brain Tumor". en. In: *arXiv:1708.05227 [cs]* (Aug. 2017). arXiv: 1708.05227. URL: http://arxiv.org/abs/1708.05227 (visited on 03/22/2019).

[43] Hojjat Salehinejad et al. "Generalization of Deep Neural Networks for Chest Pathology Classification in X-Rays Using Generative Adversarial Networks". In: *arXiv:1712.01636 [cs]* (Nov. 2017). arXiv: 1712.01636. URL: http://arxiv.org/abs/1712.01636 (visited on 03/22/2019).

[44] Xin Yi, Ekta Walia, and Paul Babyn. "Unsupervised and semi-supervised learning with Categorical Generative Adversarial Networks assisted by Wasserstein distance for dermoscopy image Classification". In: *arXiv:1804.03700 [cs]* (Apr. 2018). arXiv: 1804.03700. URL: http://arxiv.org/abs/1804.03700 (visited on 03/22/2019).

[45] Wanli Ouyang et al. "3-DeepID-Net: multi-stage and deformable deep convolutional neural networks for object detection". In: *arXiv:1409.3505 [cs]* (Sept. 11, 2014). arXiv: 1409.3505. URL: http://arxiv.org/abs/1409.3505 (visited on 05/19/2018).

[46] Mircea Cimpoi, Subhransu Maji, and Andrea Vedaldi. "Deep filter banks for texture recognition and segmentation". In: IEEE, June 2015, pp. 3828–

3836. ISBN: 978-1-4673-6964-0. DOI: 10.1109/CVPR.2015.7299007. URL: http://ieeexplore.ieee.org/document/7299007/ (visited on 05/19/2018).

[47] Bolei Zhou et al. "Learning Deep Features for Discriminative Localization". In: *arXiv:1512.04150 [cs]* (Dec. 13, 2015). arXiv: 1512.04150. URL: http://arxiv.org/abs/1512.04150 (visited on 03/01/2018).

[48] Ian Goodfellow et al. "Generative Adversarial Nets". In: *Advances in Neural Information Processing Systems 27*. Ed. by Z. Ghahramani et al. Curran Associates, Inc., 2014, pp. 2672–2680. URL: http://papers.nips.cc/paper/5423-generative-adversarial-nets.pdf (visited on 04/19/2019).

[49] Martin Arjovsky, Soumith Chintala, and Leon Bottou. "Wasserstein Generative Adversarial Networks". en. In: (), p. 10.

[50] Jonas Natten. "Generative Adversarial Networks for Improving Face Classification.pdf". PhD thesis. URL: https://brage.bibsys.no/xmlui/bitstream/handle/11250/2454822/Natten%2C%20Jonas.pdf?sequence=1 (visited on 02/22/2019).

[51] Alec Radford, Luke Metz, and Soumith Chintala. "2-Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks". In: *arXiv:1511.06434 [cs]* (Nov. 19, 2015). arXiv: 1511.06434. URL: http://arxiv.org/abs/1511.06434 (visited on 05/19/2018).

[52] Dwarikanath Mahapatra et al. "Efficient Active Learning for Image Classification and Segmentation using a Sample Selection and Conditional Generative Adversarial Network". In: *arXiv:1806.05473 [cs]* (June 14, 2018). arXiv: 1806.05473. URL: http://arxiv.org/abs/1806.05473 (visited on 02/22/2019).

[53] Dan Stowell et al. "Detection and Classification of Acoustic Scenes and Events". In: *IEEE Transactions on Multimedia* 17.10 (Oct. 2015), pp. 1733–1746. ISSN: 1520-9210, 1941-0077. DOI: 10.1109/TMM.2015.2428998. URL: http://ieeexplore.ieee.org/document/7100934/ (visited on 05/19/2018).

[54] Allen Huang and Raymond Wu. "Deep Learning for Music.pdf". In: *arXiv:1606.04930* (). URL: https://arxiv.org/pdf/1606.04930.pdf (visited on 02/23/2019).

[55] Nikhil Kotecha and Paul Young. "Generating Music using an LSTM Network". In: *arXiv:1804.07300 [cs, eess]* (Apr. 18, 2018). arXiv: 1804.07300. URL: http://arxiv.org/abs/1804.07300 (visited on 02/22/2019).