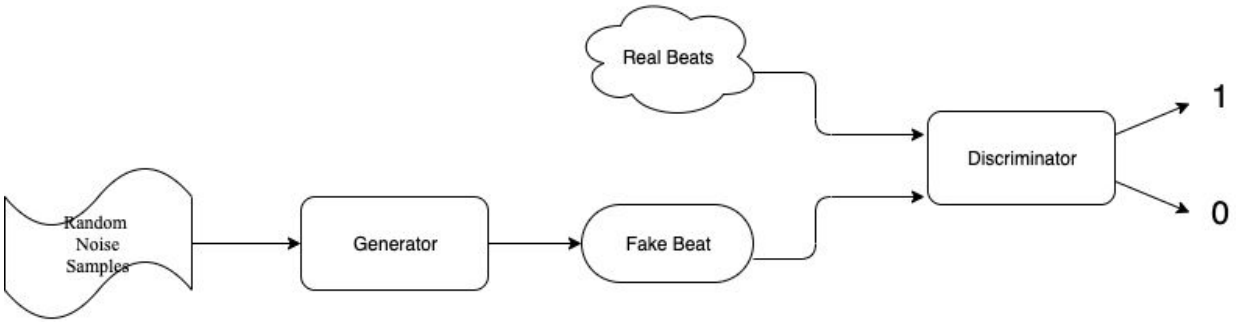
5.0 Requirements Specification

5.1 Introduction

yung gan is a command line application that will fabricate authentic-sounding hip hop beats through its unique implementation of a GAN, or Generative Adversarial Network. A vector of random noise is given as the only input to a deep network, known as the generator, where it funnels information through both recurrent and convolutional layers to create a computer generated beat in an attempt to fool another deep network, known as the discriminator. We will curate a playlist of real human-made beats from sources: Soundcloud, Youtube, and our own music. Those files will be preprocessed and stored in an SQL database along with its pertinent information. When training, those songs feed into the discriminator in batches that are combined with fake beats the generator has constructed. Over many epochs, the generator and discriminator loss should converge, indicating the training has finished. If our neural architecture is well constructed, the generator should produce beats that sound authentic to both the discriminator and to the human ear. Our system will include both a training command-line application written in Python and a generator command-line application written in Python that uses pre-trained generators included with the software to produce beats on quick command.



5.2. CSCI Component Breakdown

Our system will include both a training command-line application written in Python and a generator command-line application written in Python that uses pre-trained generators included with the software to produce beats on quick command. The features that the generator is expected to have revolves around two main inputs, the number of songs you wish to be generated and the genre of choice.

The generator command line application would work like:

generator.py <num songs> <output filepath> -g <genre> -sg <sub-genre> -fp <generator filepath>
generator.py 10 path/to/output -g rap -sg lofi

The training command line application would work like:

train.py <num epochs> <training directory> <tempo> -b <batch size> -s <save interval> -p <preprocess>
train.py 100000 path/to/training/dir -b 32 -s 100 -p true

CSCI yung gan is composed of the following CSCs:

5.2.1 Server CSC -- Amazon EC2 with GPU
5.2.1.1 Generator
5.2.1.1.1 Number of Songs
5.2.1.1.2 Output Directory
5.2.1.1.3 Genre
5.2.1.1.4 Subgenre
5.2.1.1.5 Generator Filepath
5.2.1.2 Train
5.2.1.2.1 Number of Epochs
5.2.1.2.2 Training Directory
5.2.1.2.3 Data Tempo
5.2.1.2.4 Batch Size
5.2.1.2.5 Save Interval
5.2.1.2.6 Preprocessing
5.2.2 Database CSC -- Amazon RDS for MySQL
5.2.2.1 Song Schema
5.2.2.1.1 Song Information Table
5.2.2.1.1.1 Song Name
5.2.2.1.1.2 Song Artist
5.2.2.1.1.3 Genre
5.2.2.1.1.4 Subgenre
5.2.2.1.1.5 Tempo
5.2.2.1.1.6 Scale
.

5.3 Functional Requirements by CSC

The user interface includes two command-line programs, generator and train. Generator is used for making beats from pre-trained generator networks. Train is used for training new generators from data. Generator includes functionality for generating any number of songs from any saved h5 generator deep network. Train includes functionality for taking any music dataset at any given tempo and fitting a new generator to the data. It also includes an optional preprocessing function for locking songs to the given tempo and artificially expanding the data with slight pitch and tempo randomization.

5.3.1 The generator application shall include a required input parameter for number of songs to generate

5.3.2 The generator application shall include a required input parameter for the export file path location.

5.3.3 The generator application shall include an option for sub-genre.

5.3.4 The generator application shall include an option for genre.

5.3.5 The generator application shall include an option for giving a filepath to a saved generator in h5 format.

5.3.6 The generator application shall return a help message if the user attempts to provide a generator filepath if genre or subgenre are already provided by the user

5.3.7 The generator application shall return a help message if the user attempts to provide a genre or subgenre if generator filepath is already provided.

5.3.8 The generator application shall include an option for help by showing the allowed options and mandatory parameters.

5.3.9 The generator application shall provide the user with at least one generated song per second.

5.3.10 The generator application shall save the output directly in the output destination if number of songs is given as 1.

5.3.11 The generator application shall save the outputs as a folder in the output destination if number of songs is given as more than 1.

5.3.12 The generator application shall save songs in .wav format with standard 44.1 kHz 16bit audio

5.3.13 The training application shall require the user to provide the number of training epochs for the process to undergo

5.3.14 The training application shall require the user to provide the input directory for the training set

5.3.15 The training application shall require the user to provide the tempo in beats per minute of the training set if preprocess is set to off

5.3.16 The training application shall require the user to provide their desired output tempo in beats per minute if preprocess is set to on

5.3.17 The training application shall include an option for the user to provide their desired batch size

5.3.18 The training application shall include an option for the user to provide their desired save rate

5.3.19 The training application shall include an option for the user to preprocess their dataset

5.3.20 The training application shall save the preprocessed data in its given training directory

5.3.21 The training application shall save preprocessed data in .wav format with standard 44.1 kHz 16bit audio

5.3.22 The training application shall save data samples in accordance to the save rate in .wav format with standard 44.1 kHz 16bit audio

5.3.23 The training application shall only accept training data in .wav format with standard 44.1 kHz 16bit audio

5.3.24 The training application shall provide a summary of songs loaded once loading finishes

5.3.25 The training application shall provide the loss statistics for the generator and discriminator at each epoch

5.3.26 The training application shall provide the elapsed time at each epoch

5.4 Performance Requirements by CSC

This system will perform a search of a given database, and relay information based off of the user inputs. Since the two applications within our system live solely on the command line, most performance requirements are dependent on the information the user inputs.

5.4.1 N songs

5.4.1.1 The generator application will output N songs.

5.4.2 Genre

5.4.2.1 The generator will produce N songs of the specified genre.

5.4.3 Subgenre

5.4.2.1 The generator will produce N songs of the specified genre and Subgenre.

5.4.3 Batch Size

5.4.3.1 The training application will use these many samples before it is updated.

5.4.4 Save Rate

5.4.4.1 The training application will save samples as often as the user inputs during training.

5.4.5 Preprocess

5.4.5.1 Data will either be preprocessed or not preprocessed during training

5.4.6 Tempo

5.4.6.1 The training application will run with the given tempo by the user.

5.4.7 N Epochs

5.4.7.1 The training application will use the amount of Epochs given by the user during training.

5.5 Project Environment Requirements

5.5.1 Development Environment Requirements

5.5.1.1 Minimum Requirements

CPU: 2.4GHz dual core processor.

RAM: 16 GB

Storage: 64 GB

5.5.1.2 Suggested

CPU: 2.4GHz quad core processor.

GPU: CUDA enabled NVIDIA

RAM: 64 GB

Storage: 128 GB

5.5.2 Execution Environment Requirements

5.5.2.1 Minimum Requirements

CPU: 2.4GHz dual core processor.

RAM: 16 GB

Storage: 16 GB