International Journal of Recent Technology and Engineering (IJRTE)
ISSN: 2277-3878 (Online), Volume-11 Issue-6, March 2023

# A Study on an Effective Teaching of AI using Google Colab-Based DCGAN Deep Learning Model Building for Music Data Analysis and Genre Classification

**Dong Hwa Kim**

***Abstract*: This paper deals with an effective teaching method of deep learning using theory and Python in the University. Currently, AI and related technology penetrate into all areas such as manufacturing, fashion, design, medical, novel, agriculture, as well as picture and engineering areas. These AI technologies are strongly connected with the education of universities and K-12. There are two categories of AI-related education. The first one is AI-supported education; another thing is education (teaching and learning) to understand AI. In any case, AI and its application method should be taught with theory and performed with S/W. This paper provides a method on how teachers of universities can teach deep learning well with S/W (Python) matching theory. To present the characteristics of deep learning, this paper uses DCGAN and suggests a teaching method with Google Colab easily. This paper analyzes the dataset with visuals and classifies genres to show characteristics between music and the function of deep learning for students' understanding using DCGAN and the music dataset. The results classify music genres by deep learning well.***

***Keywords*: Deep Learning, DCGAN, Music Genre, AI, Education**

## I. INTRODUCTION

The era of the 4th wave is coming into our society. For preparation at university, teachers have to teach or learn in the classroom. It means teachers have to teach after selecting one of the subjects about the 4th wave topic [1, 2, 3].

Therefore, the importance of the AI education method is increasing because AI applications, as well as basic theory and S/W, are spread in everywhere and every country widely for the preparation of the 4th wave [4, 5, 6]. The approach methods of AI education are quite different due to the country's education policy and strategy as well as teachers' capability and philosophy (teaching method and know-how). The basic necessities of AI education have already come to school, but its teaching method is not fully mature. As needs about AI education increase, learning and teaching method for students and lecturers' teaching capability is required more than ever before. As the biggest issue among current technology is AI, we should teach for the student's future jobs and on technology impact. It is the best leverage and important role for society as educational institutes [7, 8, 9].

There are some tools, such as Python, Pytorch, and Matlab, as implementation S/W of AI to teach AI [10], but Python is the best one to teach at university because free of charge, there are many references, and many communities to communicate when they have a problem [11].

When we teach or learn Python, we can have an idea Jupiter notebook or Google Colab as tools. If you have a GPU in your Lab, the Jupyter notebook is a good teaching method as the S/W system (S/W+PC). However, students and teachers consider many hard works because of bugs or running time. It is not an easy job for teachers, and many students as beginners or not developer level in the classroom. Teachers and students must have a pressure of running speed without GPU [11, 15].

Herein, Google Colab-based Python deep learning teaching and learning approach is the best tool for teaching and learning in the classroom because it is a convenient and easy-to-use way to operate deep learning or machine learning with access to GPU with free.

Additionally, also you can have many advantages:

- Speed up training using Google Colab's GPU,
- Using Google Colab's system to save Google drive,
- Save your model programming,
- Easy to teach (or learn) because of step by step process.

When we use the Jupyter notebook on PC, there are many jobs for installing, preparing, and making an environment for teaching in the classroom. It means teachers must have many and much works for teaching activities. However, Google Colab is so easy because it just turns on and starts step by step (Python basic).

This paper provides experience and research materials on how we can build easy teaching (or learning) and program deep learning through DCGAN programming and music data. Basically, music data is waveform as sound, but it can be transferred to visuals and graphs through Python. We can also analyze the characteristics of the music genre by waveform and visual patterns through deep learning. This paper uses DCGAN to analyze those and provide programs for deep learning model building of DCGAN [11, 20, 21].

**Manuscript received on 10 November 2022 | Revised Manuscript received on 10 December 2022 | Manuscript Accepted on 15 March 2023 | Manuscript published on 30 March 2023.**

***Correspondence Author (s)**
**Dong Hwa Kim***, NDT Center, Seoul National Science and Technology University, S. Korea. Email: koreahucare@gmail.com, ORCID ID: https://orcid.org/0000-0002-0528-6736

© The Authors. Published by Blue Eyes Intelligence Engineering and Sciences Publication (BEIESP). This is an open access article under the CC-BY-NC-ND license http://creativecommons.org/licenses/by-nc-nd/4.0/

## II. STUDY BACKGROUND

### A. The Advantage of Deep Learning Study Using Music Data Analysis

The music data is composed of waveform and sound. So, the students and researchers can understand some parts of engineering, such as frequency, the relation between frequency and image shape, and the relation between waveform and sound well because it is a basic engineering area. As music is also one of the cultural areas, combining AI and culture is a very good idea for the future paradigm of the job, and students can understand well the relationship between culture and music [15, 19].

The current study of tagging or recommendation methods using deep learning (or machine learning) is useful for research and new business. By using AI (deep learning), students understand the application method of AI more than any other way. Music is one of the best cultural tools to illustrate a human being's life and care for the mental situation of everybody. AI (Artificial Intelligence) is the best convenient and useful tool for music analysis and its characteristics (Macharla Vaibhavi, 2021). Music genres and styles are so many depend on race (culture, custom, lifestyle) and person.

When animals, flowers, plants as well as humans the world hear music, they feel the emotion and give influence others and around them, and a piece of music moves them. Music is a communication system with sound, rhythm, harmony, and others. Therefore, it is very valuable to analyze and express why and how music impacts everything. If we analyze these music characteristics and styles on why and what impact on others, that will be good and valuable research [6].

Music looks like a very complicated emotional expression tool. Still, it is composed of sound and silence, rhythm, pitch (melody and harmony), rhythm (meter and tempo, and articulation), dynamics, and the qualities of timbre and texture. Basically, music has many behavioral and cultural aspects of studying and modifying for creation. Therefore, MIR (Music Information Retrieval) has the best interdisciplinary materials for studying data analysis and AI applications, as well as music sources and culture [17, 19].

Through this research, we can develop new content and create a new paradigm for the future society. It is very useful for students to teach deep learning-based music analysis for application methods and deep learning understanding as well as cultural patterns because the analysis of music by AI has many advantages [16, 12].

### B. MSD Dataset of Music

The Million Song Dataset is the largest and most diverse free audio data for a million music tracks for researchers [18].

**Table I: Size Comparison with Other Datasets [4].**

| Dataset | # Songs / Samples | Audio |
|---|---|---|
| RWC | 465 | Yes |
| CAL500 | 502 | No |
| GZTAN genre | 1,000 | Yes |
| USPOP | 8,752 | No |
| Swat10K | 10,870 | No |
| Magnatagatune | 25,863 | Yes |
| OMRAS2 | 50,000 | No |
| Musi CLEF | 2,00,000 | Yes |
| MSD | 10,00,000 | N0 |

It provides to encourage research on algorithms and a reference dataset for research and helps new researchers. Teachers explain the status of the dataset and its characteristics, as well as getting method. Teachers must express the importance of the dataset and how and why they should make labeling. Of course, teachers must say that they have to get and build their data. Dataset about them cannot be imported like goods.

The lecturer has to explain the characteristics and the importance of the dataset for the connection of deep learning.

In this teaching experience, we used dataset that proved for building of DCGAN. It has one million songs and data set cluster: Second Hand Songs dataset (cover songs); musi Xmatch dataset (lyrics); Last.fm dataset (song-level tags and similarity); Taste Profile subset (user data); thisismyjam-to-MSD mapping (more user data); tagtraum genre annotations (genre labels); Top MAGD dataset (more genre labels).

The MSD is started as a collaborative project between The Echo Nest and Lab ROSA supported by the NSF (George, Tzanetakis, et el., 2002).

### C. GTZAN Dataset

It is a very famous and useful dataset for analysis and AI application research.

This dataset was produced by G. Tzanetakis and P. Cook, supporting by NSF.

That is, G. Tzanetakis and P. Cook published the article on genre classification with the title "Musical genre classification of audio signals" in IEEE Transactions on Audio and Speech Processing [13, 14].

The GTZAN has the most public music data for music genre research. It is gathered during 2000-2001 through CD, radio, microphone record

(https://www.kaggle.com/datasets/andradaolteanu/gtzan-dataset-music-genre-classification). Its dataset has 1000 audio tracks totally with a 30-second duration. It has 10 genres, totally each with 100 tracks. All the tracks have 22050Hz Mono 16-bit audio files in wave format. The genres for this analysis are Disco, Hip-hop, Jazz, Metal, Pop, Reggae, Rock, Blues, Classical, and Country.

Additionally, there are other datasets that this paper does not describe. This paper uses this dataset of 30 seconds for analysis.

## III. VISUALIZATION OF DATA

We can analyze the music datasets through visualizations such as frequency, charts, plots, graphics, and animations. Through these visual results and AI tools, users can create ideas, illustrate well, analyze, and apply them easily. Types of data visualizations have tables, charts, graphs, histograms, scatter plots, heat maps, and tree maps. The purpose of this paper is to see and provide information analysis know-how on how to analyze music datasets and apply AI to these datasets. To analyze, this paper uses Google Colab-based Python and DCGAN for music genre and how to code DCGAN to teach the coding method and the structure of DCGAN.The lecturer should explain why we must show the visualization of music data.


© Copyright: All rights reserved.

That is, we can show the characteristics of the dataset as the image because deep learning can classify genres using the image of the dataset. of course, the lecturer explains the coding method with Python for images expressed in Figure 1 to Figure 9. If they can extend the zoom of the image, they have to show the waveform on the big screen for the student's understanding.

### A. Importing Libraries

The lecturer uses imported libraries to be loaded our dataset using pandas:

```
df = pd.read_csv
("/kaggle/input/gtzan-dataset-music-genre-classification/Data/features_3_sec.csv")
df.head()
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import scipy
import os
import pickle
import librosa
import librosa.display
from IPython.display import Audio
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
import tensorflow as tf
from tensorflow import keras
```

The above sentence (program) is the path of the program used in this paper. Therefore, the lecture must set this path for their situation. Almost case, the program cannot be runed because of this path error ("/kaggle/input/gtzan-dataset-music-genre-classification/Data/features_3_sec.csv").

### B. Audio Libraries

Basically, to analyze music and audio datasets, users use the Librosa tool programmed by the Python package.

The Librosa offers the building blocks necessary to create MIR (Music Information Retrieval) systems.

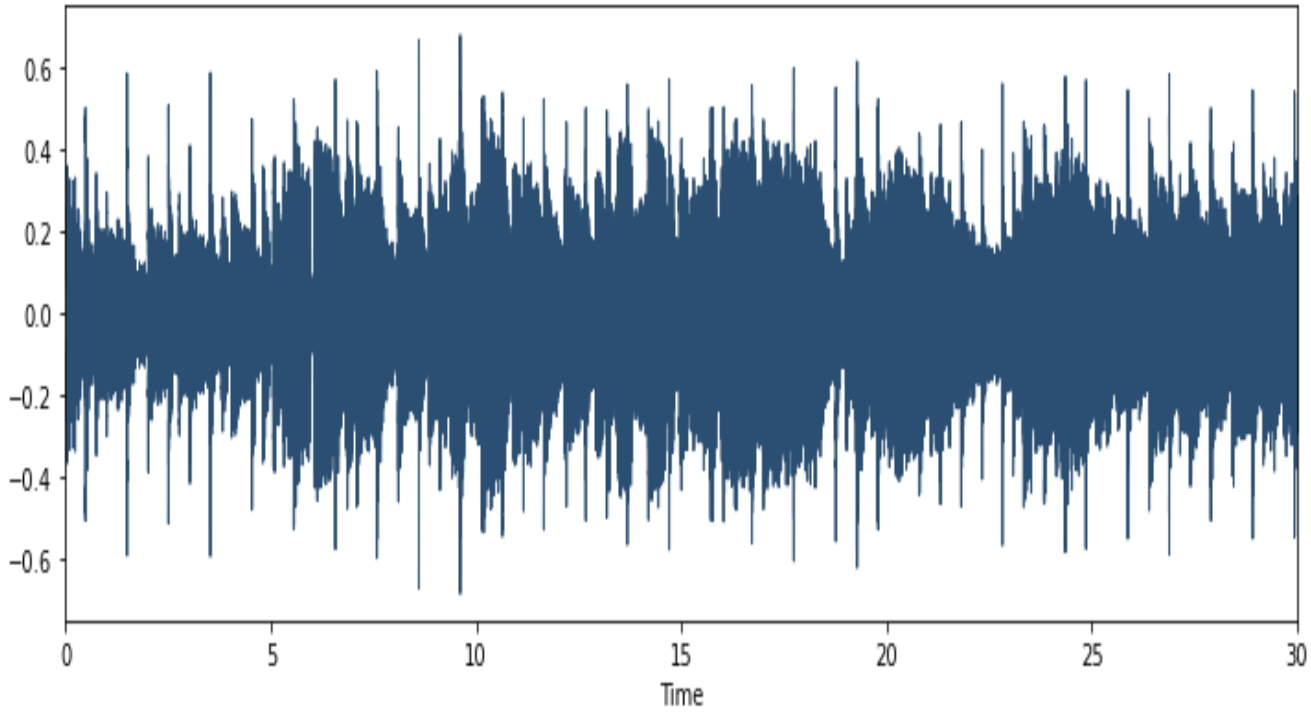


**Figure 1. Plotting Raw Wave Files**

We can extract certain key features from the audio samples such as Tempo, Chroma Energy Normalized, Mel-Frequency Cepstral Coefficients, Spectral Centroid, Spectral Contrast, Spectral Rolloff, and Zero Crossing Rate using this Librosa. The code is below:

```
import IPython
IPython.display.Audio(data,rate=sr)
```

### C. Plotting Raw Wave of Audio 1

Figure 1 shows the waveforms of music data for 30 seconds used in this paper. It is a visual representation of sound as time on the x-axis and amplitude on the y-axis. This visual graph can quickly scan the audio data and visually compare and contrast which genres might be more similar than others.

Using this visual graph, students can understand the characteristics of waveform well depending genre of music.

### D. Plotting Raw Wave of Audio 2

Figure 2 represents the Python code and visualization of waveforms of audio 2 music data of 30 seconds used in this paper. The waveform differs slightly from Figure 1, which means the waveform depends on the music style.


© Copyright: All rights reserved.

Audio 1

```
[ ] idx = np.random.randint(0,len(data))
    music_names = data["Unnamed: 0"][idx]
    path = r"Verified_Normed/"+music_names
    y, sampling_rate = librosa.load(path)
    waveplot(y, sampling_rate,music_names)
    spectrogram(y, sampling_rate, music_names)
    Audio(path)
```

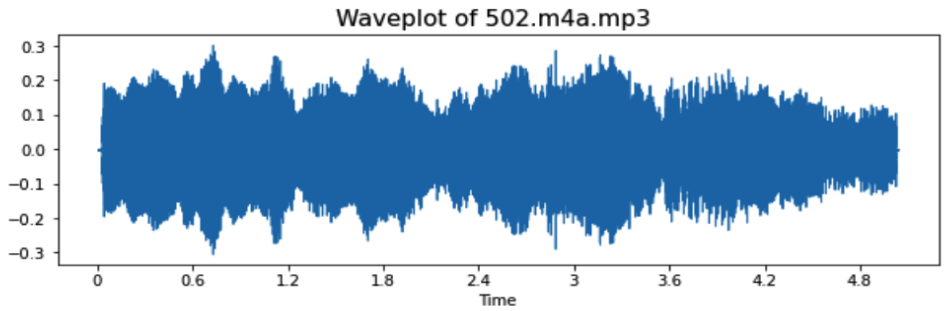


**Figure 2. Data Visualization of Audio 2**

### E. Plotting Raw Wave of Audio 3

Figure 3 is the Python code and waveform visualization of audio 3. These results show that the code contents and the results of the visualization of each audio are different depending on the music.

### F. Audio Spectrogram

An audio spectrogram is one of the visual ways of illustrating the signal of loudness over time at various frequencies in the loud waveform.

By using this waveform, students can see not only the energy structures of music but also the characteristics of music genres. This audio spectrum is quite different depending on the music genre as well as cultures. It means we can create a new paradigm of music and content as this reference. With this spectrum, we can plot 3D of a dataset of music, and the resulting data is more detailed information about music.

Audio 2

```
[ ] idx = np.random.randint(0,len(data))
    music_names = data["Unnamed: 0"][idx]
    path = r"Verified_Normed/"+music_names
    y, sampling_rate = librosa.load(path)
    waveplot(y, sampling_rate,music_names)
    spectrogram(y, sampling_rate, music_names)
    Audio(path)
```

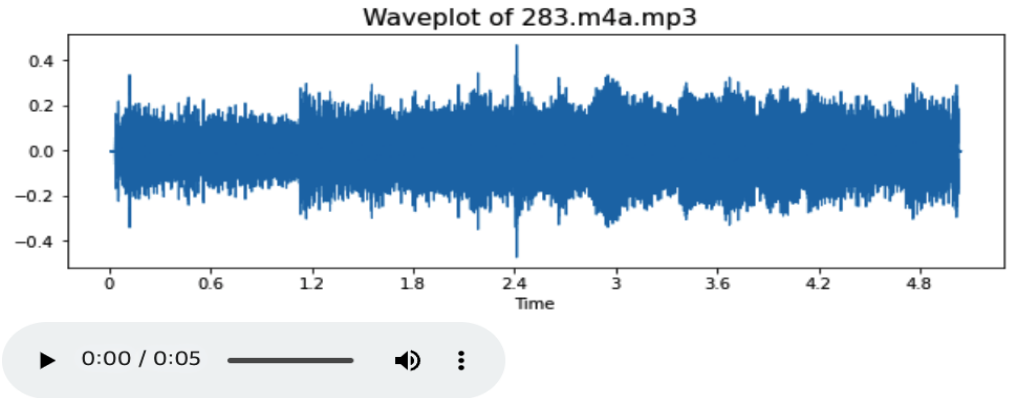


**Figure 3. The Graph of Audio 3 and Coding**

Retrieval Number: 100.1/ijrte.E73510111523
DOI: 10.35940/ijrte.E7351.0311623
Journal Website: www.ijrte.org



Published By:
Blue Eyes Intelligence Engineering and Sciences Publication (BEIESP)
© Copyright: All rights reserved.

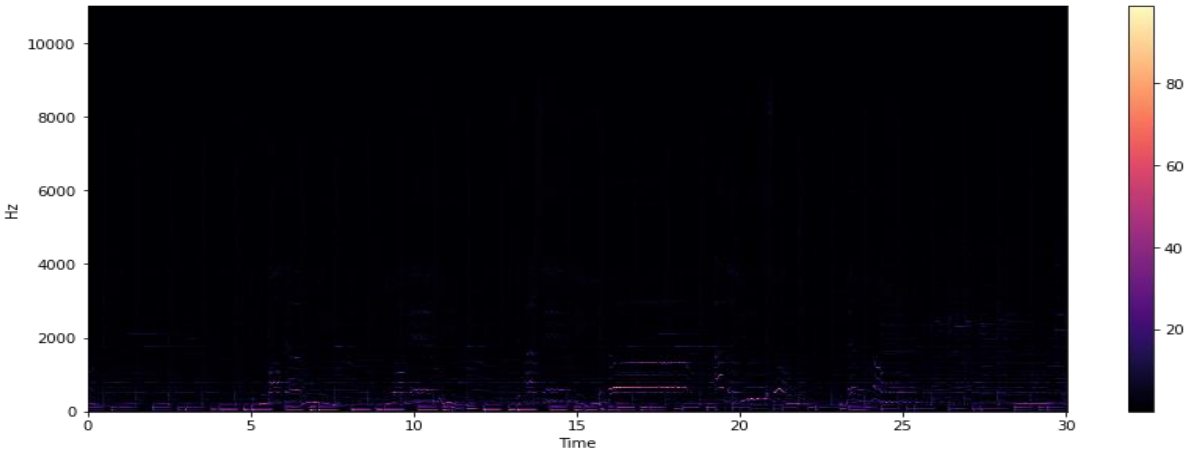


**Figure 4: Spectrogram of Audio 3**

Figure 4 illustrates the frequency and time for the spectrogram.

Here is the code:

```
stft=librosa.stft(data)
stft_db=librosa.amplitude_to_db(abs(stft))
plt.figure(figsize=(14,6))
librosa.display.specshow(stft,sr=sr,x_axis='time',y_axis='hz'
)
plt.colorbar()
```

```
stft=librosa.stft(data)
stft_db=librosa.amplitude_to_db(abs(stft))
plt.figure(figsize=(14,6))
librosa.display.specshow(stft_db,sr=sr,x_axis='time',y_axis=
'hz')
plt.colorbar()
```

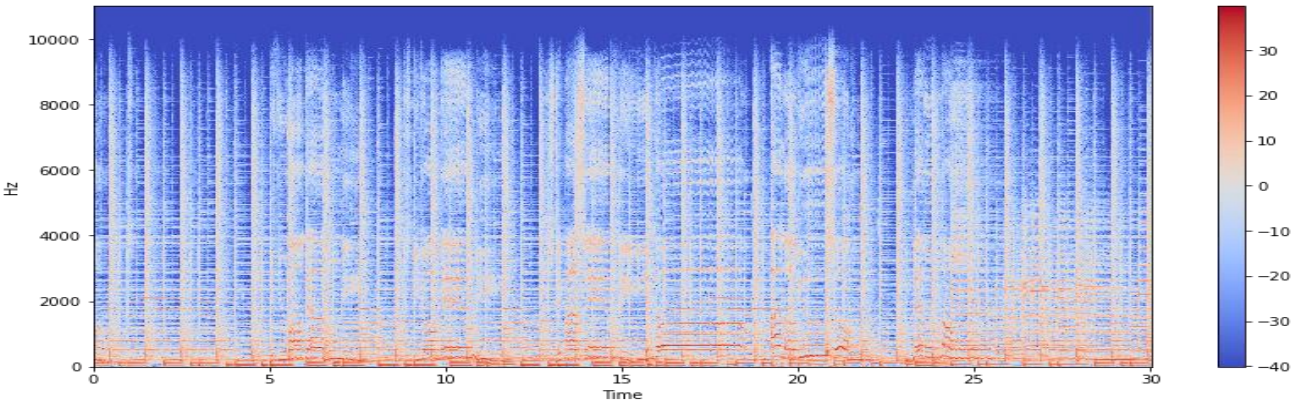


**Figure 5. Color bar of Audio 1**

Figures 5, 6, and 7 show the visualization characteristics of the color bar of music frequency. The results are different depending on the music. We can see the genre characteristics of music and classify genre by this spectrogram and AI. This color bar provides image information well to classify genres by DCGAN.

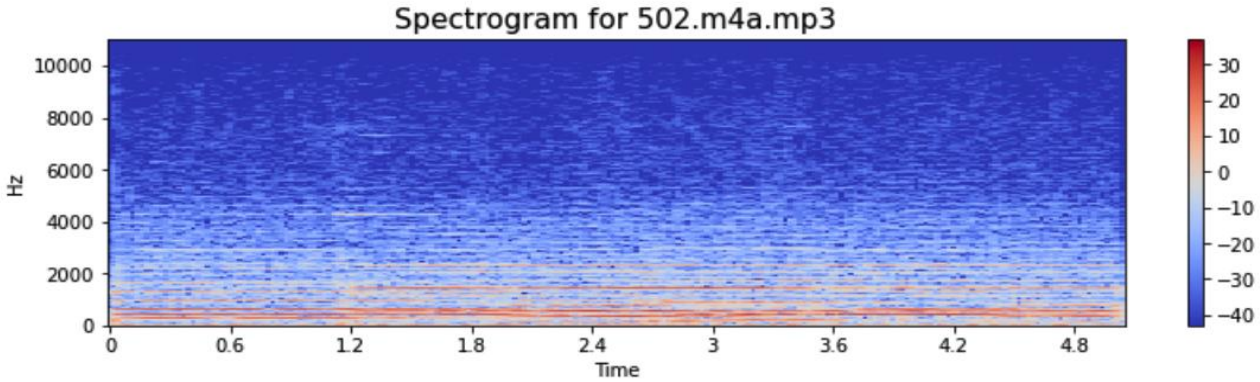


**Figure 6. Color bar of Audio 2**


© Copyright: All rights reserved.

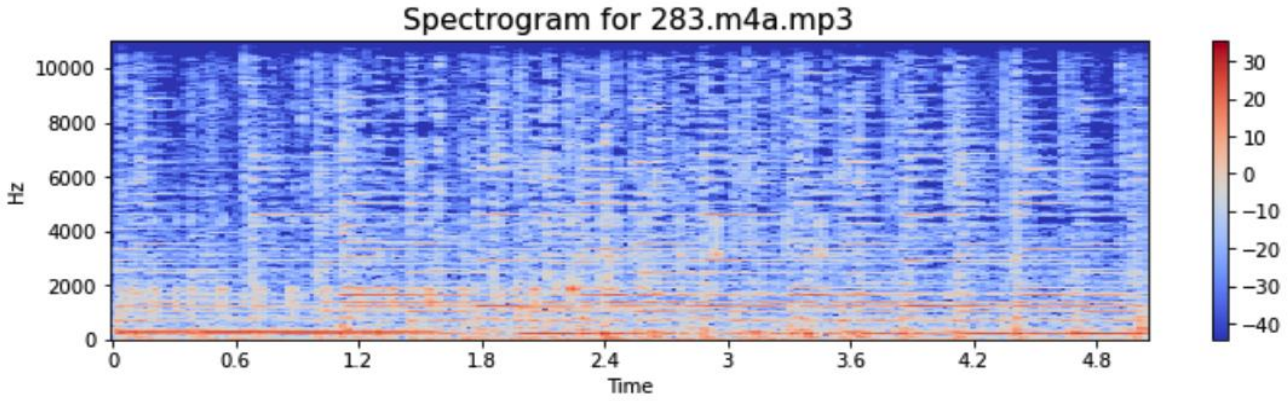


**Figure 7. Color bar of Audio 3**

### G. Spectral Roll-Off

Spectral Rollof is one of the ways of representing total spectral energy of music by frequency, as shown in Figure 8.

The coding method is below:

```
spectral_rolloff=librosa.feature.spectral_rolloff(data+0.01,sr=sr)[0]
plt.figure(figsize=(14,6))
librosa.display.waveplot(data,sr=sr,alpha=0.4,color="#2B4F72")
```

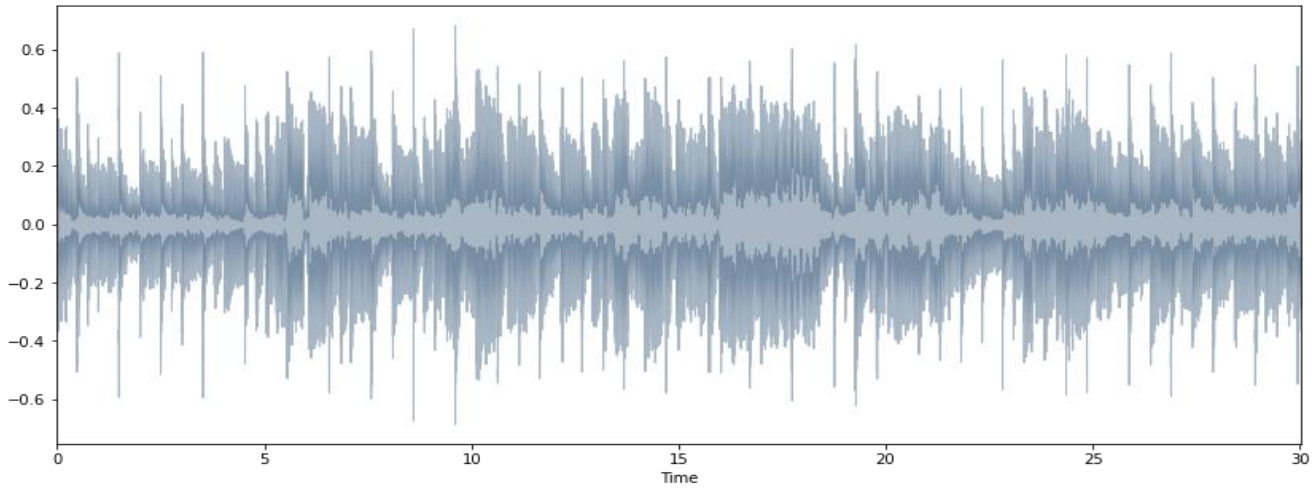


**Figure 8. Spectral Roll off of Music Energy**

### H. Chroma Feature

The chroma feature is one of the expressions for frequency pitch. The lecturers can analyze this method, which introduces this frequency analysis method to show effective music characteristics.

This chroma feature method is a powerful tool for analysis of music features that these pitches can provide meaningfully category and the equal-tempered scale.

This paper obtains harmonic and melodic characteristics of music through the property of chroma features. Of course, we can see timbre and instrumentation.

```
import librosa.display as lplt
chroma = librosa.feature.chroma_stft(data,sr=sr)
plt.figure(figsize=(14,6))
lplt.specshow(chroma,sr=sr,x_axis="time",y_axis="chroma",cmap="coolwarm")
plt.colorbar()
plt.title("Chroma Features")
plt.show()
```

Retrieval Number: 100.1/ijrte.E73510111523
DOI: 10.35940/ijrte.E7351.0311623
Journal Website: www.ijrte.org



Published By:
Blue Eyes Intelligence Engineering and Sciences Publication (BEIESP)
© Copyright: All rights reserved.

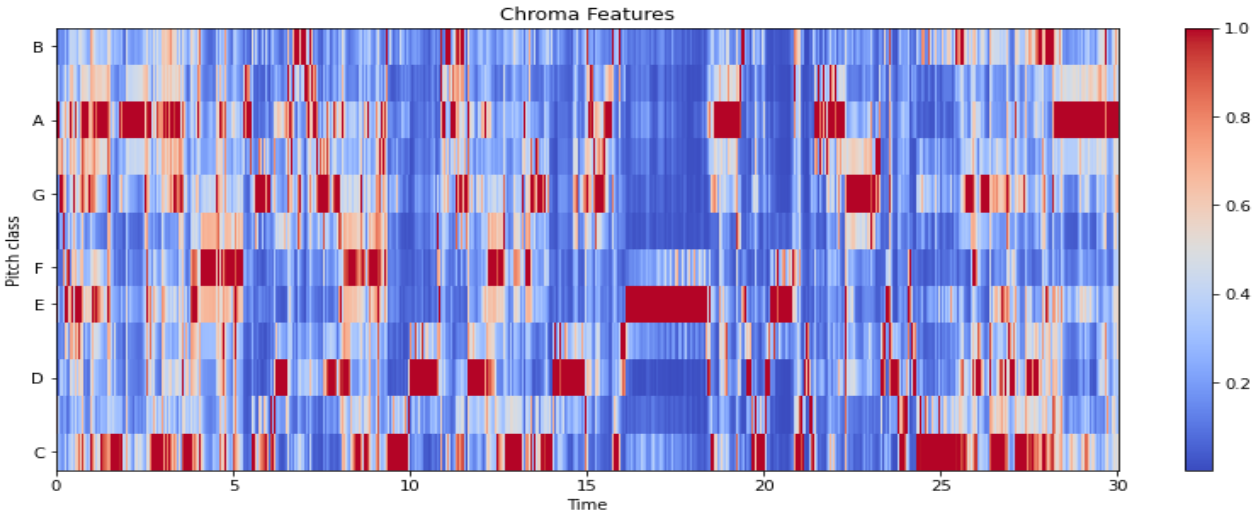


**Figure 9. Chroma Feature**

### I. Zero Crossing Rate

Zero crossing is to illustrate how many times the signal of music signals up and down the zero line of the X-axis (zero-crossing rate) as shown in figure 10.

The rate at which zero-crossings occur is a simple measure of the frequency content of a signal. Zero-crossing rate is a measure of the number of times in a given time interval/frame that the amplitude of the speech signals passes through a value of zero. It illustrates the characteristics of genre in music.

```
start=1000
end=1200
plt. figure (figsize=(12,4))
plt.plot(data[start:end],color="#2B4F72")
plt.grid()
```

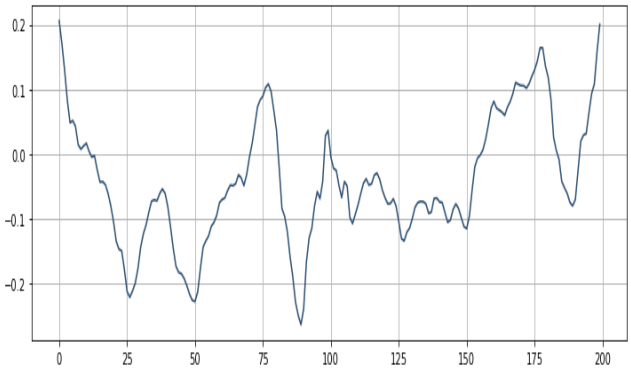


**Figure 10. Zero Crossing Rate**

### J. Feature Extraction

To train data, we must pre-process data (data dealing with). It means that we must make on the last column that is 'label' and encode it with the function Python (LabelEncoder() of sklearn preprocessing) as below.

```
class_list=df.iloc[:,-1]
converter=LabelEncoder()
y=converter.fit_transform(class_list)
y
print(df.iloc[:,:-1])
```

### K. Scaling the features

To connect deep learning, we must standardize data using a standard scaler for unit variance.

Usually, the standard score of sample x is calculated as: z = (x - u) / s

```
from sklearn.preprocessing import StandardScaler
fit=StandardScaler()
X=fit.fit_transform(np.array(df.iloc[:,:-1],dtype=float))
```

### L. Dividing Training and Testing Dataset

To connect the AI model, we divide training data and test data as below:

```
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.33)
len(y_test)
len(y_train)
```

This philosophy is quite important for the building of 4th industrial revolution infrastructure because manpower is the best solution for that.

## IV. DCGAN MODEL EVALUATION

As the DCGAN is one structure of deep learning, it learns by comparing the generator (generating fake data) and discriminator (comparing, as shown in Figure 11.

### A. The Basic Learning and Structure of DCGAN

The structure of DCGAN is shown in Figure 11, and its basic learning theory is as follows [19, 20, 21]:

$$\min_G \max_D V(D,G) = E_{x,P_{data}} log[D(x)] + E_{x,P_z} log[1 - D(z)] \quad (1)$$


© Copyright: All rights reserved.

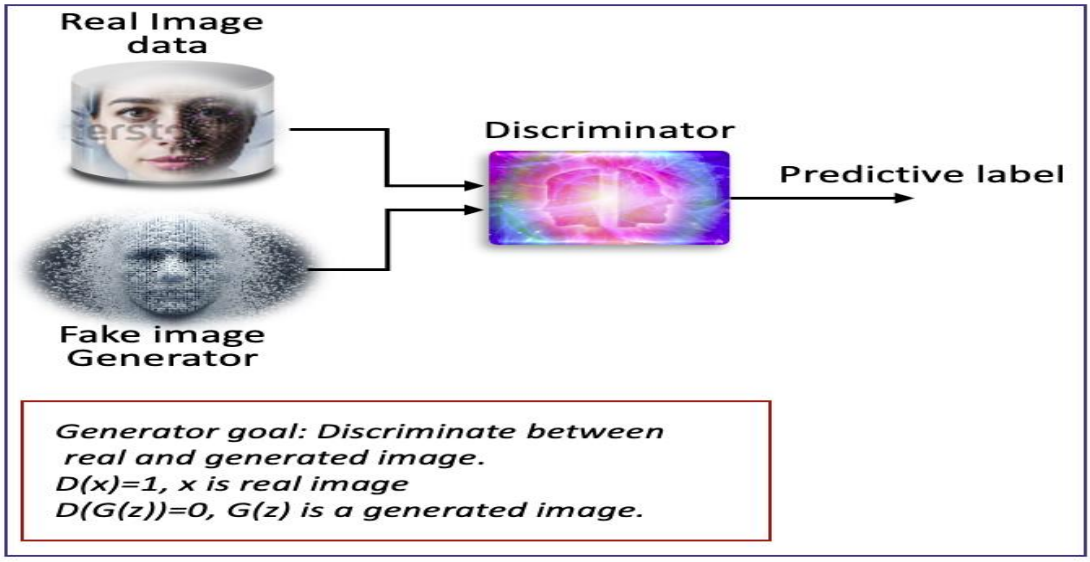


**Figure 11 (a). The Principle of DCGAN**

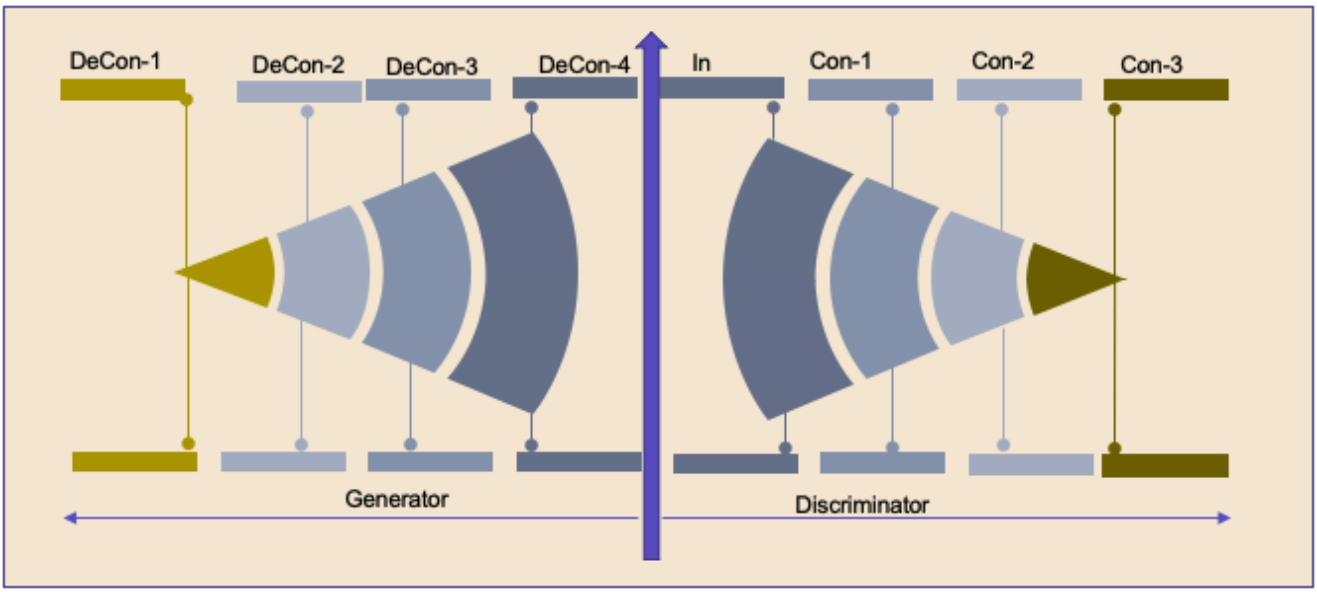


**Figure 11 (b). DCGAN Structure**

The generator has a role to generate fake images, and the discriminator has the function of deciding how much it is close fake image data to real image data through learning.

The fake generator must produce updated image data to avoid exact figuring out of the discriminator, and the discriminator should figure out fake image data by comparing fake image data and real image data. The lecturer explains the function of these two modules.

### B. The principle of running

The lecture teaches the meaning of equation (1) and explains the roles of fake generators and discriminators. And the lecture introduces the code below:

Of course, the lecturer explains the function of Sequential, Dense, Conv2D, Leaky Relu, Lambda, and other codes listed below. The lecturers had better show with DCGAN model figure for students' understanding.

```
import os
import cv2
import pandas as pd
import numpy as np
from numpy import expand_dims
from numpy import zeros
from numpy import ones
from numpy import asarray
from numpy.random import randn
from numpy.random import randint
from sklearn.preprocessing import OneHotEncoder
from sklearn.preprocessing import MinMaxScaler
from tensorflow.keras.optimizers import Adam
from keras.models import Model,Sequential
from keras.layers import Input
from keras.layers import Dense
from keras.layers import Reshape
from keras.layers import Flatten
from keras.layers import Conv2D
from keras.layers import Conv2DTranspose
from keras.layers import LeakyReLU
from keras.layers import Dropout
from keras.layers import Lambda
from keras.layers import Activation
from keras.metrics import Precision,Recall
import matplotlib.pyplot as plt
from keras import backend
```

Retrieval Number: 100.1/ijrte.E73510111523
DOI: 10.35940/ijrte.E7351.0311623
Journal Website: www.ijrte.org



Published By:
Blue Eyes Intelligence Engineering and Sciences Publication (BEIESP)
© Copyright: All rights reserved.

```
[2] data = pd.read_csv("features_30_sec.csv")
    data = data.sample(frac=1).reset_index(drop=True)
```

```
[3] target = data.label.values
    data.drop(["filename","length","label"],axis=1,inplace=True)

    features = data.values
    oe_enc = OneHotEncoder()
    target = oe_enc.fit_transform((target).reshape(-1,1)).toarray()
```

```
[4] scaler = MinMaxScaler()
    features = scaler.fit_transform(features)
```

```
[5] # define the standalone discriminator model
    def define_discriminator(n_inputs=57,n_classes = 10):
        in_dim = Input(shape=n_inputs)
        fe = Dense(64, activation='relu', kernel_initializer='he_uniform')(in_dim)
        fe = Dense(128, activation='relu', kernel_initializer='he_uniform')(fe)
        fe = Dense(256, activation='relu', kernel_initializer='he_uniform')(fe)
        c_out_layer = Dense(n_classes)(fe)
        c_out_layer = Activation('softmax')(c_out_layer)
        c_model = Model(in_dim, c_out_layer)
        c_model.compile(loss='categorical_crossentropy', optimizer=Adam(lr=0.0002, beta_1=0.5), metrics=['accuracy'])

        d_out_layer = Dense(1)(fe)
        d_out_layer =Activation('softmax')(d_out_layer)
        # define and compile unsupervised discriminator model
        d_model = Model(in_dim, d_out_layer)
        d_model.compile(loss='binary_crossentropy', optimizer=Adam(lr=0.0002, beta_1=0.5),metrics=['accuracy'])
        return d_model, c_model
```

```
[6] # define the standalone generator model
    def define_generator(latent_dim, n_outputs=57):
        model = Sequential()
        model.add(Dense(15, activation='relu', kernel_initializer='he_uniform', input_dim=latent_dim))
        model.add(Dense(n_outputs, activation='linear'))
        return model
```

```
[7] # define the combined generator and discriminator model, for updating the generator
    def define_gan(generator, discriminator):
        # make weights in the discriminator not trainable
        discriminator.trainable = False
        # connect them
        model = Sequential()
        # add generator
        model.add(generator)
        # add the discriminator
        model.add(discriminator)
        # compile model
        model.compile(loss='binary_crossentropy', optimizer='adam')
        return model
```

```
[8] # generate n real samples with class labels
    def generate_real_samples(n):
        ix = np.random.randint(0, features.shape[0], n)
        # select images and labels
        X, labels = features[ix], target[ix]
        # generate class labels
        y = np.ones((n, 1))
        return [X,labels], y
```

```
[9] # generate points in latent space as input for the generator
    def generate_latent_points(latent_dim, n):
        # generate points in the latent space
        x_input = np.random.randn(latent_dim * n)
        # reshape into a batch of inputs for the network
        x_input = x_input.reshape(n, latent_dim)
        return x_input
```

```
[10] # use the generator to generate n fake examples, with class labels
    def generate_fake_samples(generator, latent_dim, n):
        # generate points in latent space
        x_input = generate_latent_points(latent_dim, n)
        # predict outputs
        X = generator.predict(x_input)
        # create class labels
        y = np.zeros((n, 1))
        return X, y
```

```
[11] # evaluate the discriminator and plot real and fake points
    def summarize_performance(epoch, generator, discriminator,c_model, latent_dim, n=100):
        # prepare real samples
        X, y_real = generate_real_samples(n)
        x_real,Y= X
        # evaluate discriminator on real examples
        _, acc_real = discriminator.evaluate(x_real, y_real, verbose=0)
        # prepare fake examples
        x_fake, y_fake = generate_fake_samples(generator, latent_dim, n)
        # evaluate discriminator on fake examples
        _, acc_fake = discriminator.evaluate(x_fake, y_fake, verbose=0)

        _, acc_class = c_model.evaluate(x_real, Y, verbose=0)
        # summarize discriminator performance
        print('Epoch : {}, Real acc : {}, Fake acc : {}, Classification acc : {}'.format(epoch, acc_real, acc_fake,acc_class))
        return acc_class
```

```
[12] # train the generator and discriminator
    def train(g_model, d_model,c_model, gan_model, latent_dim, n_epochs=1000, n_batch=128, n_eval=10):
        # determine half the size of one batch, for updating the discriminator
        half_batch = int(n_batch / 2)
        num_steps = int(features.shape[0]/n_batch)
        # manually enumerate epochs
        for i in range(n_epochs):
            for j in range(num_steps):
                # prepare real samples
                X, y_real = generate_real_samples(half_batch)
                x_real,Y = X
                c_loss, c_acc = c_model.train_on_batch(x_real, Y)
                # prepare fake examples
                x_fake, y_fake = generate_fake_samples(g_model, latent_dim, half_batch)
                # update discriminator
                d_loss1,_ = d_model.train_on_batch(x_real, y_real)
                d_loss2,_ = d_model.train_on_batch(x_fake, y_fake)
                # prepare points in latent space as input for the generator
                x_gan = generate_latent_points(latent_dim, n_batch)
                # create inverted labels for the fake samples
                y_gan = np.ones((n_batch, 1))
                # update the generator via the discriminator's error
                g_loss = gan_model.train_on_batch(x_gan, y_gan)
            print('>%d, c[%.3f,%.0f], d[%.3f,%.3f], g[%.3f]' % (i+1, c_loss, c_acc*100, d_loss1, d_loss2, g_loss))
            # evaluate the model every n_eval epochs
            if (i+1) % n_eval == 0:
                acc_1 = summarize_performance(i, g_model, d_model,c_model, latent_dim)
                accuracy_classification.append(acc_1)
```

```
[13] accuracy_classification = []
    # size of the latent space
    latent_dim = 100
    # create the discriminator
    discriminator,c_model = define_discriminator()
    # create the generator
    generator = define_generator(latent_dim)
    # create the gan
    gan_model = define_gan(generator, discriminator)
    # train model
    train(generator, discriminator,c_model, gan_model, latent_dim)
```



Published By:
Blue Eyes Intelligence Engineering
and Sciences Publication (BEIESP)
© Copyright: All rights reserved.

```
/usr/local/lib/python3.7/dist-packages/keras/optimizer_v2/adam.py:105: UserWarning: The `lr` argument is deprecated, use `learning_rate` instead.
  super(Adam, self).__init__(name, **kwargs)
>1, c[2.395,14], d[0.516,0.750], g[0.749]
>2, c[2.249,22], d[0.430,0.544], g[0.923]
>3, c[2.236,22], d[0.354,0.478], g[1.136]
>4, c[2.230,19], d[0.312,0.398], g[1.295]
>5, c[2.187,25], d[0.249,0.289], g[1.502]
>6, c[2.201,19], d[0.206,0.256], g[1.648]
>7, c[2.170,25], d[0.174,0.235], g[1.842]
>8, c[2.075,34], d[0.144,0.201], g[1.869]
>9, c[2.073,36], d[0.132,0.197], g[1.925]
>10, c[2.121,16], d[0.125,0.186], g[2.157]
Epoch : 9, Real_acc : 1.0, Fake_acc : 0.0, Classification_acc : 0.38999998569488525
>11, c[1.978,47], d[0.143,0.182], g[2.038]
>12, c[2.003,38], d[0.120,0.189], g[2.212]
>13, c[1.855,48], d[0.126,0.146], g[2.342]
>14, c[1.932,36], d[0.133,0.161], g[2.265]
>15, c[1.731,58], d[0.144,0.170], g[2.358]
>16, c[1.784,31], d[0.172,0.169], g[2.250]
>17, c[1.809,47], d[0.184,0.203], g[2.027]
>18, c[1.709,39], d[0.249,0.238], g[1.904]
>19, c[1.539,52], d[0.255,0.304], g[1.908]
>20, c[1.489,52], d[0.420,0.267], g[2.085]
Epoch : 19, Real_acc : 1.0, Fake_acc : 0.0, Classification_acc : 0.46000000834465027
>21, c[1.525,61], d[0.299,0.234], g[2.207]
>22, c[1.542,50], d[0.328,0.302], g[2.167]
>23, c[1.664,45], d[0.376,0.278], g[2.114]
>24, c[1.460,55], d[0.536,0.335], g[1.861]
>25, c[1.400,62], d[0.615,0.697], g[1.535]
>26, c[1.430,56], d[0.809,0.718], g[1.203]
>27, c[1.438,48], d[0.761,0.645], g[1.281]
>28, c[1.295,66], d[0.751,0.492], g[1.278]
>29, c[1.399,61], d[0.892,0.579], g[1.523]
>30, c[1.312,50], d[0.913,0.550], g[1.368]
Epoch : 29, Real_acc : 1.0, Fake_acc : 0.5799999833106995
```

**Figure 12. Python Code Running Process of Google Colab for DCGAN**

```
[14] plt.figure(figsize=(15,8))
     plt.title("Accuracy")
     num_epochs = [i for i in range(len(accuracy_classification))]
     plt.plot(num_epochs,accuracy_classification,"-.")
     plt.show()
```

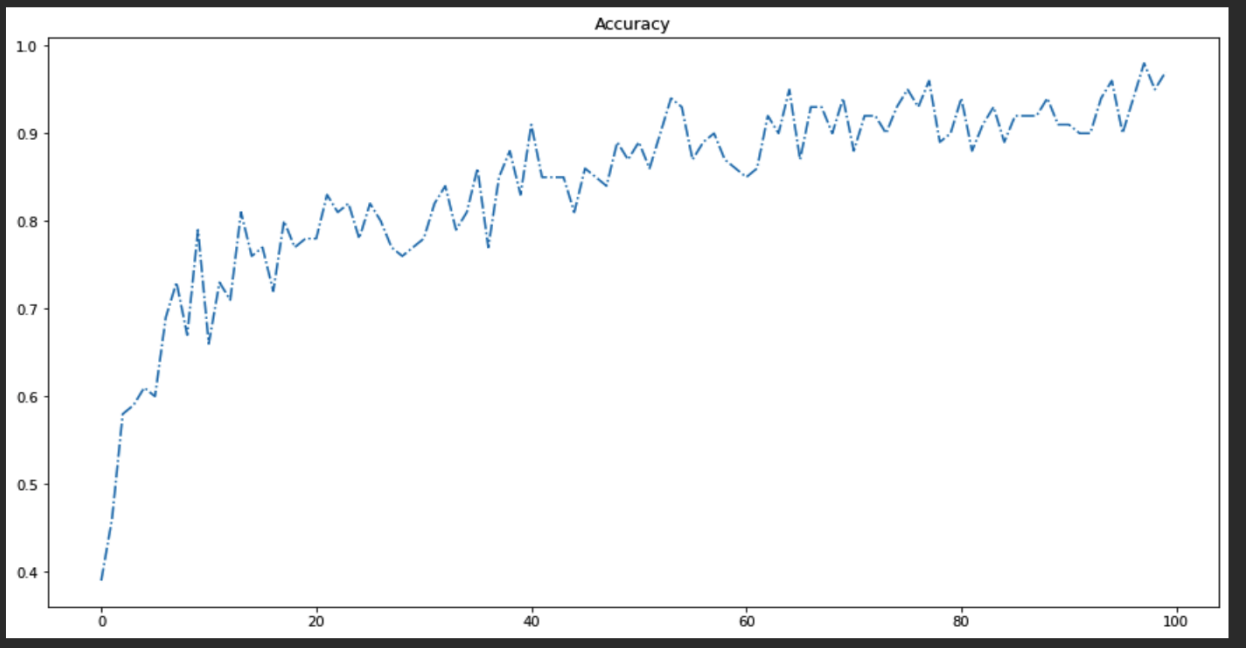


**Figure 13 (a). Learning Processing of DCGAN**

Retrieval Number: 100.1/ijrte.E73510111523
DOI: 10.35940/ijrte.E7351.0311623
Journal Website: www.ijrte.org



Published By:
Blue Eyes Intelligence Engineering and Sciences Publication (BEIESP)
© Copyright: All rights reserved.

## Visulization

```
[ ] data = pd.read_csv("MeanCategoryRating/MeanAffectRatingsUSA.csv")
    data.head()
```

| | Unnamed: 0 | arousal | attention | certainty | commitment | dominance | enjoyment | identity | obstruction | safety | valence | familiarity |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 01 - Tree Dance&I.mp3 | 4.3333 | 4.6667 | 5.0000 | 5.0000 | 4.0000 | 4.3333 | 4.6667 | 4.0000 | 5.6667 | 4.6667 | 4.3333 |
| 1 | 04F4xlWSFh0_18.mp3 | 7.3333 | 4.6667 | 6.6667 | 3.6667 | 6.6667 | 2.6667 | 6.6667 | 3.0000 | 4.0000 | 2.6667 | 6.6667 |
| 2 | 04F4xlWSFh0_75.mp3 | 8.0000 | 2.6667 | 4.0000 | 2.6667 | 8.3333 | 2.6667 | 6.0000 | 6.0000 | 3.3333 | 4.0000 | 7.0000 |
| 3 | 05UApr2UQHQ_227.mp3 | 4.3333 | 4.0000 | 4.0000 | 3.6667 | 2.3333 | 5.3333 | 2.6667 | 4.6667 | 4.6667 | 5.0000 | 2.0000 |
| 4 | 073TNjLc070_40.mp3 | 5.3333 | 5.0000 | 5.0000 | 5.3333 | 6.6667 | 5.6667 | 5.3333 | 4.3333 | 4.6667 | 5.3333 | 4.6667 |

**Figure 13 (b). Learning Processing of DCGAN**

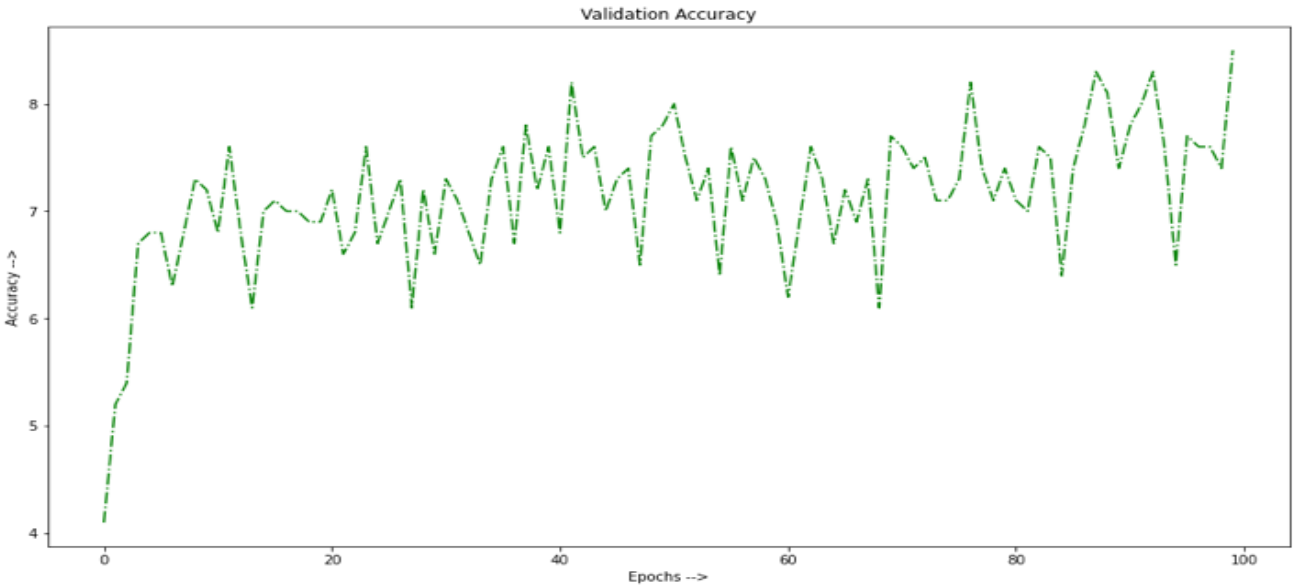


**Figure 14 (a). Learning Accuracy of DCGAN**

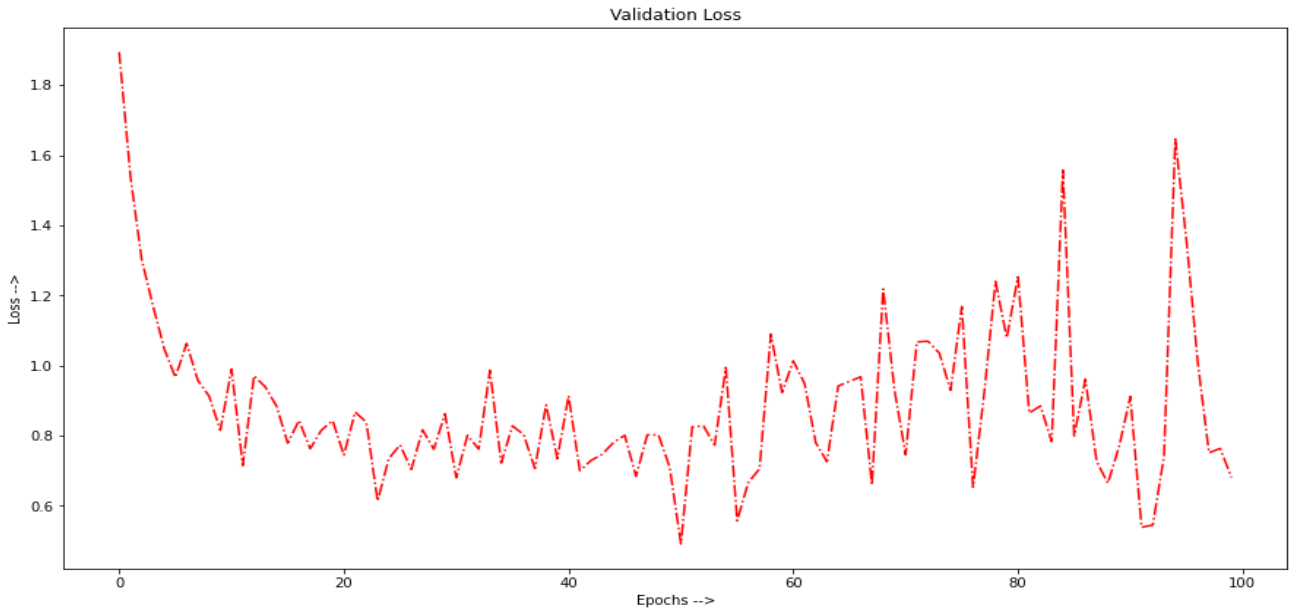


**Figure 14 (b). Learning Accuracy of DCGAN**


© Copyright: All rights reserved.

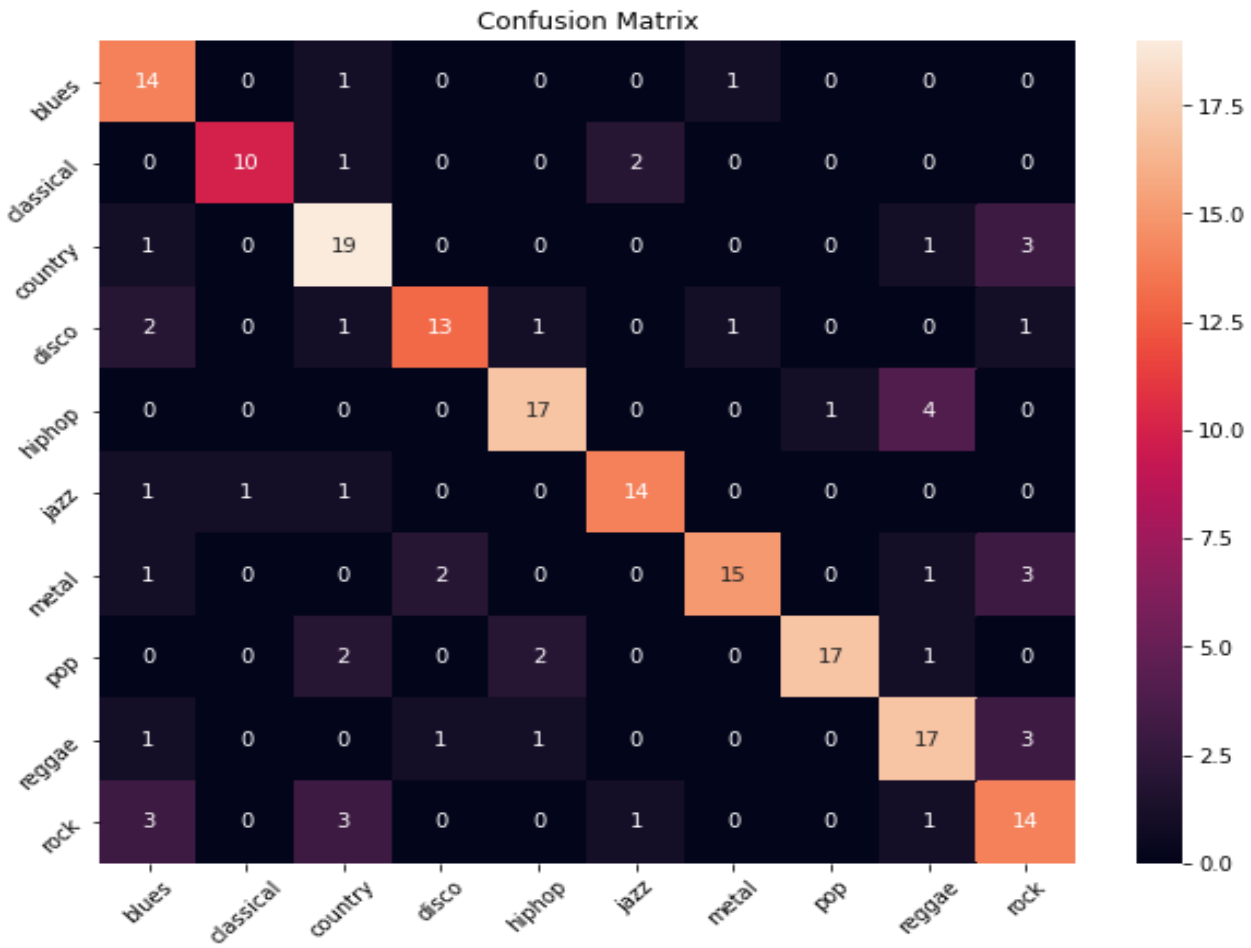


**Figure 15. Confusion Matrix by DCGAN to Classification**

Figure 12 shows the code list and learning status in PC. Figure 13 is the learning processing of DCGAN used in this paper.

Figure 14 represents the learning accuracy of DCGAN. Figure 15 is the confusion matrix of genre classification by DCGAN.

## V. CONCLUSION

Currently, so many agencies and universities are interesting in deep learning and its application because of the core technology of the 4th wave. The application and its theory are so widely depending on the deep learning structure and its application areas. So, the coding method is not easy for beginners because they must match the theory, application area, model, and code method. The developer can understand well because they are already experts.

However, students' understanding is quite different depending on the teaching method. Even though the contents do not have high technology in deep learning, teaching skill is quite important for students in the university as much as high technology. This paper provides materials and methods for how lecturers can teach well students and beginners through the author's teaching experience.

## ACKNOWLEDGMENT

These works were supported by the National Research Foundation of Korea (NRF) grant funded by the Korean government (MSIT) (No. 2021R1F1A1056145). The author thanks to support of the Korean government (MSIT).

## DECLARATION

| | |
|---|---|
| Funding/ Grants/ Financial Support | Yes, received funds/grants/financial support for this article. This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korean government (MSIT) (No. 2021R1F1A1056145). The author thanks to supporting of the Korean government (MSIT). |
| Conflicts of Interest/ Competing Interests | No conflicts of interest to the best of our knowledge. |
| Ethical Approval and Consent to Participate | The author must submit a statement that the study does not require ethical approval and consent to participate with evidence. |
| Availability of Data and Material/ Data Access Statement | Not relevant. |
| Authors Contributions | I am only the sole author of the article. |


© Copyright: All rights reserved.

## REFERENCES


1. Upendra Shardanand (1995). Algorithms for Automating (Word of Mouth). CHI'95 MOSAIC of creativity. 201-217.
2. 2.https://www.who.int/news/item/09-02-2022-ensuring-artificial-intelligence-(ai)-technologies-for-health-benefit-older-people.
3. https://www.itu.int/en/ITU-T/AI/Pages/default.aspx.
4. https://bernardmarr.com/how-is-ai-used-in-education-real-world-examples-of-today-and-a-peek-into-the-future/
5. David Karandish (2021). https://thejournal.com/articles/2021/06/23/7-benefits-of-ai-in-education.aspx.
6. Lisa Plitnichenko (2020). https://elearningindustry.com/5-main-roles-artificial-intelligence-in-education.
7. Alex McFarland (2022). https://www.unite.ai/10-best-ai-tools-for-education/.
8. KoreaHerald (2020). https://www.koreaherald.com/common/newsprint.php?ud=20201120000655.
9. 9.https://www.analyticsinsight.net/top-10-ai-tools-for-researchers-to-make-their-work-easy-in-2021/
10. Anirudh V K, 2002. https://www.spiceworks.com/tech/artificial-intelligence/articles/best-ai-tools/.
11. Python basic. https://colab.research.google.com/github/data-psl/lectures2020/blob/master/notebooks/01_python_basics.ipynb
12. Xiaowen Cheng, "Time-frequency Analysis of Musical Rhythm. Journal of Mathematics and Music," pp. 1–21, 2007.
13. http://millionsongdataset.com/
14. Thierry Bertin-Mahieux, Daniel P.W. Ellis, "The million song dataset," 2012.
15. Macharla Vaibhavi, "Music Genre Classification using Neural Networks with Data Augmentation," Foundation for Scientific Research & Technological Innovation, pp. 21-37, 2021.
16. Juhan Nam, Keunwoo Choi, "Deep Learning for Audio-Based Music Classification and Tagging. IEEE Signal Processing Magazine, Vol. 43-51, 2019
17. Pasi Saari. Semantic models of musical mood: Comparison between crowd-sourced and curated edition tags. https://www.researchgate.net/publication/257151618.
18. Van Loi Nguyen, "A New Recognition Method for Visualizing Music Emotion. International Journal of Electrical and Computer Engineering," IJECE, Vol. 7, No. 3, pp. 1246~1254, 2017. [CrossRef]
19. George, Tzanetakis, et el., "Musical Genre Classification of Audio Signals," IEEE Transactions on speech and audio processing, Vol. 10, No. 5, pp. 293-302, 2002. [CrossRef]
20. https://www.javatpoint.com/python-programming-with-google-colab.
21. Sungho Suh (et el., "Generative Oversampling Method for Imbalanced Data on Bearing Fault Detection and Diagnosis," Appl. Sci., Vol.9, pp. 1-16, 2019. [CrossRef]


## AUTHORS PROFILE

**Dong Hwa Kim,** Ph.D.: Dept. of Computational Intelligence and Systems Science, Interdisciplinary Graduate School of Science and Engineering (AI Application for Automatic control), TIT (Tokyo Institute of Technology), Tokyo, Japan. He worked at the Hanbat National University (Dean, Prof., S. Korea); Prof. at Electrical Power and Control Eng. Adama Science and Tech. Uni., Ethiopia; TDTU, Vietnam. He has experience in many universities overseas as Prof. He was NCP of EU-FP7 (EU-Framework Program, ICT). He had a keynote speaker at several international conferences and universities. He has 200 papers in journals and conferences. He is reviewing IEEE and other's journals. He is currently a researcher at the Seoul national university of S&T. He published many books and papers such as Innovation tuning based on biotechnology (USA, Dec. 2017), 4th wave Status and preparation of Visegrad Group Country (Germany, 2019), How to They Education in the Famous Univ. (2019), Africa and 4th Wave: Will it risk or Chance? (Amazon, 2020), How to teach and Learn AI (Outskirt Press, USA, Aug. 2022), A Study on Reinforcement of Self-Directed Learning Using Controlling Face Emotion (Paper, Jan. 2022), Advanced Lectures for PID Controller of Nonlinear System in Python (IJRTE, March 2021), Dynamic Decoupling and Intelligent Optimal PID Controller Tuning Multivariable Qua-drones (IJRTE (Scopus), Dec. 2021), Failure Prediction of Wind Turbine using Neural Network and Operation Signal (IJRTE, Dec. 2021), and 200 papers.

- Home page: www.worldhumancare.wixsite.com/kimsite
- Research citations: https://www.researchgate.net/profile/Dong_Kim53

**Disclaimer/Publisher's Note:** The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of the Blue Eyes Intelligence Engineering and Sciences Publication (BEIESP)/ journal and/or the editor(s). The Blue Eyes Intelligence Engineering and Sciences Publication (BEIESP) and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.


*© Copyright: All rights reserved.*