```
import numpy as np
import pandas as pd
import os

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import tensorflow as tf
from sklearn.model_selection import train_test_split
import librosa.display
import IPython.display as ipd
from IPython.display import Audio
from keras.models import Sequential
from keras.layers import Dense, Dropout, Activation, Flatten
from keras.layers import Conv2D, MaxPooling2D

sound_df = pd.read_csv(r"C:\Users\Shaivya\Desktop\Data\
features_3_sec.csv")
sound_df.head()

            filename  length  chroma_stft_mean  chroma_stft_var
rms_mean  \
0  blues.00000.0.wav   66149          0.335406         0.091048
0.130405
1  blues.00000.1.wav   66149          0.343065         0.086147
0.112699
2  blues.00000.2.wav   66149          0.346815         0.092243
0.132003
3  blues.00000.3.wav   66149          0.363639         0.086856
0.132565
4  blues.00000.4.wav   66149          0.335579         0.088129
0.143289

    rms_var  spectral_centroid_mean  spectral_centroid_var  \
0  0.003521             1773.065032          167541.630869
1  0.001450             1816.693777           90525.690866
2  0.004620             1788.539719          111407.437613
3  0.002448             1655.289045          111952.284517
4  0.001701             1630.656199           79667.267654

   spectral_bandwidth_mean  spectral_bandwidth_var  ...  mfcc16_var  \
0              1972.744388           117335.771563  ...   39.687145
1              2010.051501            65671.875673  ...   64.748276
2              2084.565132            75124.921716  ...   67.336563
3              1960.039988            82913.639269  ...   47.739452
4              1948.503884            60204.020268  ...   30.336359

   mfcc17_mean  mfcc17_var  mfcc18_mean  mfcc18_var  mfcc19_mean
mfcc19_var  \
0    -3.241280   36.488243     0.722209   38.099152    -5.050335
```

```
33.618073
1    -6.055294   40.677654     0.159015   51.264091    -2.837699
97.030830
2    -1.768610   28.348579     2.378768   45.717648    -1.938424
53.050835
3    -3.841155   28.337118     1.218588   34.770935    -3.580352
50.836224
4     0.664582   45.880913     1.689446   51.363583    -3.392489
26.738789

   mfcc20_mean  mfcc20_var  label
0    -0.243027   43.771767  blues
1     5.784063   59.943081  blues
2     2.517375   33.105122  blues
3     3.630866   32.023678  blues
4     0.536961   29.146694  blues

[5 rows x 60 columns]
```

```
sound_df['label'].value_counts()
```

```
label
blues        1000
jazz         1000
metal        1000
pop          1000
reggae       1000
disco         999
classical     998
hiphop        998
rock          998
country       997
Name: count, dtype: int64
```

```
sound_df.dtypes
```

```
filename                       object
length                          int64
chroma_stft_mean              float64
chroma_stft_var               float64
rms_mean                      float64
rms_var                       float64
spectral_centroid_mean        float64
spectral_centroid_var         float64
spectral_bandwidth_mean       float64
spectral_bandwidth_var        float64
rolloff_mean                  float64
rolloff_var                   float64
zero_crossing_rate_mean       float64
zero_crossing_rate_var        float64
```

```
harmony_mean              float64
harmony_var               float64
perceptr_mean             float64
perceptr_var              float64
tempo                     float64
mfcc1_mean                float64
mfcc1_var                 float64
mfcc2_mean                float64
mfcc2_var                 float64
mfcc3_mean                float64
mfcc3_var                 float64
mfcc4_mean                float64
mfcc4_var                 float64
mfcc5_mean                float64
mfcc5_var                 float64
mfcc6_mean                float64
mfcc6_var                 float64
mfcc7_mean                float64
mfcc7_var                 float64
mfcc8_mean                float64
mfcc8_var                 float64
mfcc9_mean                float64
mfcc9_var                 float64
mfcc10_mean               float64
mfcc10_var                float64
mfcc11_mean               float64
mfcc11_var                float64
mfcc12_mean               float64
mfcc12_var                float64
mfcc13_mean               float64
mfcc13_var                float64
mfcc14_mean               float64
mfcc14_var                float64
mfcc15_mean               float64
mfcc15_var                float64
mfcc16_mean               float64
mfcc16_var                float64
mfcc17_mean               float64
mfcc17_var                float64
mfcc18_mean               float64
mfcc18_var                float64
mfcc19_mean               float64
mfcc19_var                float64
mfcc20_mean               float64
mfcc20_var                float64
label                      object
dtype: object
```

```
sound_df.shape
```

```
(9990, 60)
```

```
sound_df.describe()
```

```
        length  chroma_stft_mean  chroma_stft_var      rms_mean
rms_var  \
count   9990.0       9990.000000      9990.000000  9990.000000
9.990000e+03
mean   66149.0          0.379534         0.084876     0.130859
2.676388e-03
std        0.0          0.090466         0.009637     0.068545
3.585628e-03
min    66149.0          0.107108         0.015345     0.000953
4.379535e-08
25%    66149.0          0.315698         0.079833     0.083782
6.145900e-04
50%    66149.0          0.384741         0.085108     0.121253
1.491318e-03
75%    66149.0          0.442443         0.091092     0.176328
3.130862e-03
max    66149.0          0.749481         0.120964     0.442567
3.261522e-02

       spectral_centroid_mean  spectral_centroid_var
spectral_bandwidth_mean  \
count             9990.000000           9.990000e+03
9990.000000
mean              2199.219431           4.166727e+05
2241.385959
std                751.860611           4.349644e+05
543.854449
min                472.741636           8.118813e+02
499.162910
25%               1630.680158           1.231961e+05
1887.455790
50%               2208.628236           2.650692e+05
2230.575595
75%               2712.581884           5.624152e+05
2588.340505
max               5432.534406           4.794119e+06
3708.147554

       spectral_bandwidth_var  rolloff_mean  ...  mfcc16_mean
mfcc16_var  \
count            9.990000e+03   9990.000000  ...  9990.000000
9990.000000
mean             1.182711e+05   4566.076592  ...     1.448240
49.988755
std              1.013505e+05   1642.065335  ...     5.735149
34.442816
```

```
min                1.183520e+03    658.336276  ...   -26.850016
1.325786
25%                4.876553e+04   3378.311110  ...    -2.227478
29.584894
50%                8.996072e+04   4631.377892  ...     1.461623
41.702393
75%                1.585674e+05   5591.634521  ...     5.149752
59.274619
max                1.235143e+06   9487.446477  ...    39.144405
683.932556

       mfcc17_mean   mfcc17_var  mfcc18_mean   mfcc18_var  mfcc19_mean
\
count  9990.000000  9990.000000  9990.000000  9990.000000  9990.000000

mean     -4.198706    51.962753     0.739943    52.488851    -2.497306

std       5.677379    36.400669     5.181313    38.177120     5.111799

min     -27.809795     1.624544   -20.733809     3.437439   -27.448456

25%      -7.951722    29.863448    -2.516638    29.636197    -5.734123

50%      -4.443021    42.393583     0.733772    41.831377    -2.702366

75%      -0.726945    61.676964     3.888734    62.033906     0.514246

max      34.048843   529.363342    36.970322   629.729797    31.365425


        mfcc19_var  mfcc20_mean   mfcc20_var
count  9990.000000  9990.000000  9990.000000
mean     54.973829    -0.917584    57.322614
std      41.585677     5.253243    46.444212
min       3.065302   -35.640659     0.282131
25%      30.496412    -4.004475    30.011365
50%      43.435253    -1.030939    44.332155
75%      65.328602     2.216603    68.210421
max    1143.230591    34.212101   910.473206

[8 rows x 58 columns]
```

```
audio_sample = r"C:\Users\Shaivya\Desktop\Data\genres_original\pop\
pop.00055.wav"
data, sr = librosa.load(audio_sample)
print(type(data), type(sr))
```

```
<class 'numpy.ndarray'> <class 'int'>
```

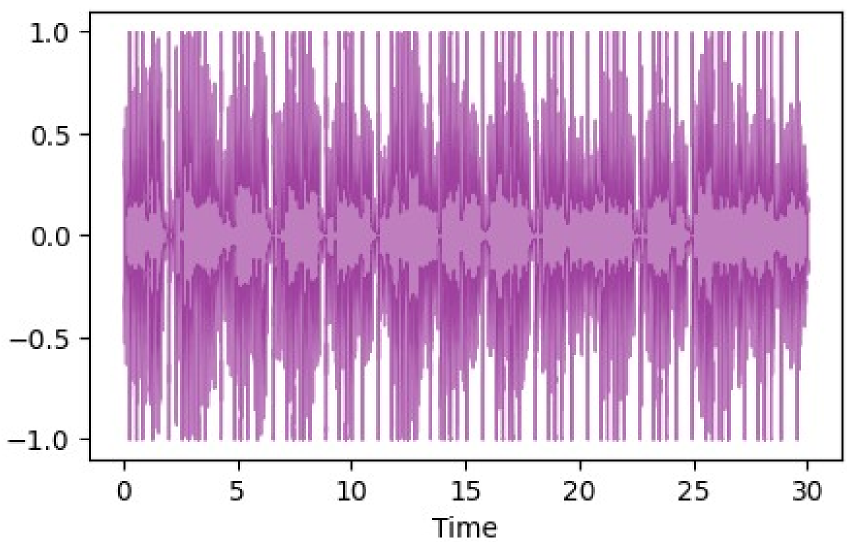
```
librosa.load(audio_sample, sr = None)
```

```
(array([ 0.00384521,  0.02346802,  0.07144165, ...,  0.04241943,
        -0.10992432, -0.07177734], dtype=float32),
 22050)
```

```
librosa.load(audio_sample, sr = None)
```

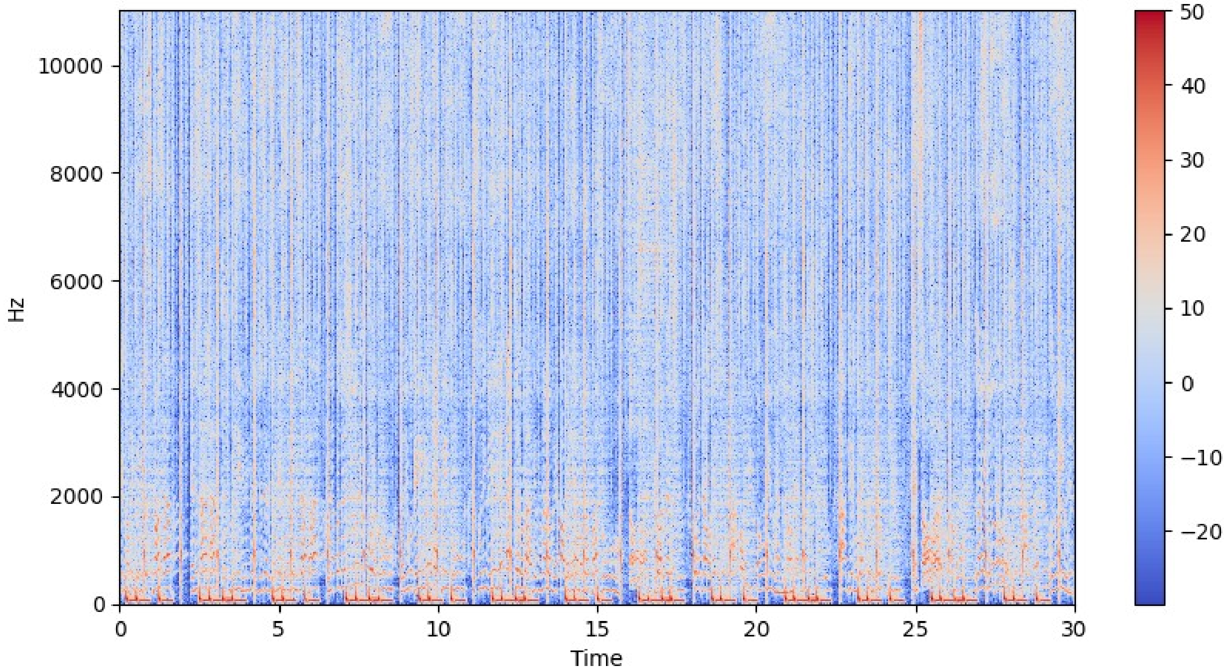
```
(array([ 0.00384521,  0.02346802,  0.07144165, ...,  0.04241943,
        -0.10992432, -0.07177734], dtype=float32),
 22050)
```

```
plt.figure(figsize = (5,3))
librosa.display.waveshow(data, color = "purple", alpha = 0.5)
plt.show()
```

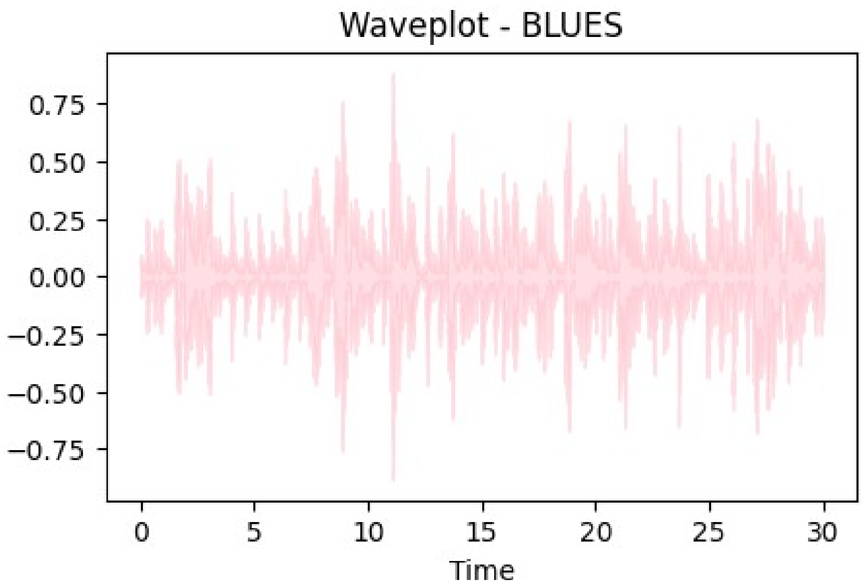


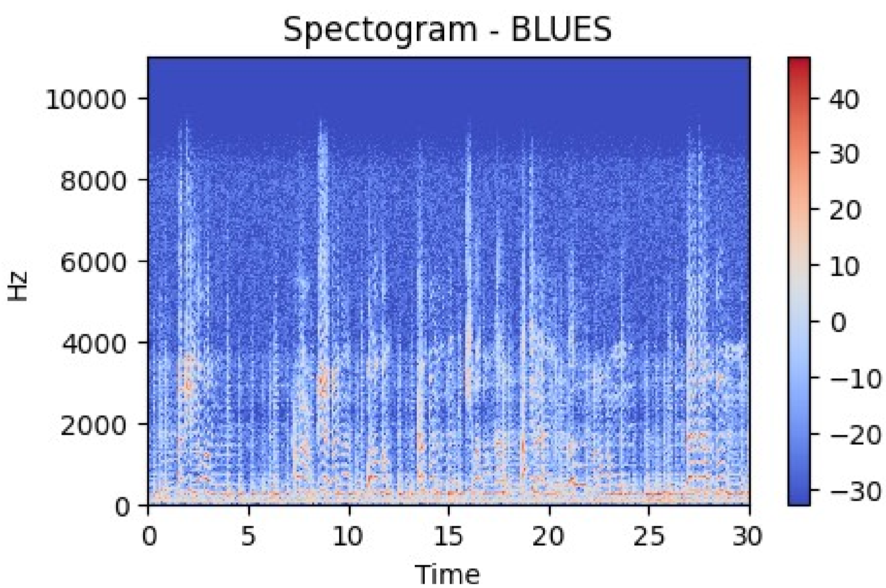

```
stft = librosa.stft(data)
stft_db = librosa.amplitude_to_db(abs(stft))
plt.figure(figsize = (10, 5))
librosa.display.specshow(stft_db, sr = sr, x_axis = 'time', y_axis =
'hz')
plt.colorbar()
```

```
<matplotlib.colorbar.Colorbar at 0x243bc1727d0>
```




```
# 1. BLUES
audio_blues = r"C:\Users\Shaivya\Desktop\Data\genres_original\blues\
blues.00015.wav"
data_blues, sr = librosa.load(audio_blues)
plt.figure(figsize=(5, 3))
librosa.display.waveshow(data_blues, sr = sr, color = "pink", alpha =
0.5)
plt.title('Waveplot - BLUES')

# Spectrogram
stft = librosa.stft(data_blues)
stft_db = librosa.amplitude_to_db(abs(stft))
plt.figure(figsize = (5, 3))
librosa.display.specshow(stft_db, sr = sr, x_axis = 'time', y_axis =
'hz')
plt.title('Spectogram - BLUES')
plt.colorbar()

# Playing audio
ipd.Audio(audio_blues)

<IPython.lib.display.Audio object>
```

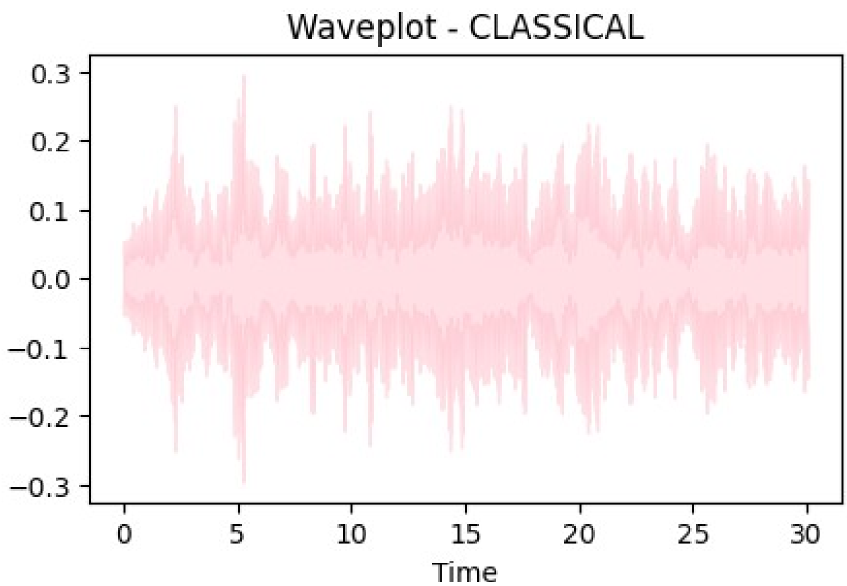


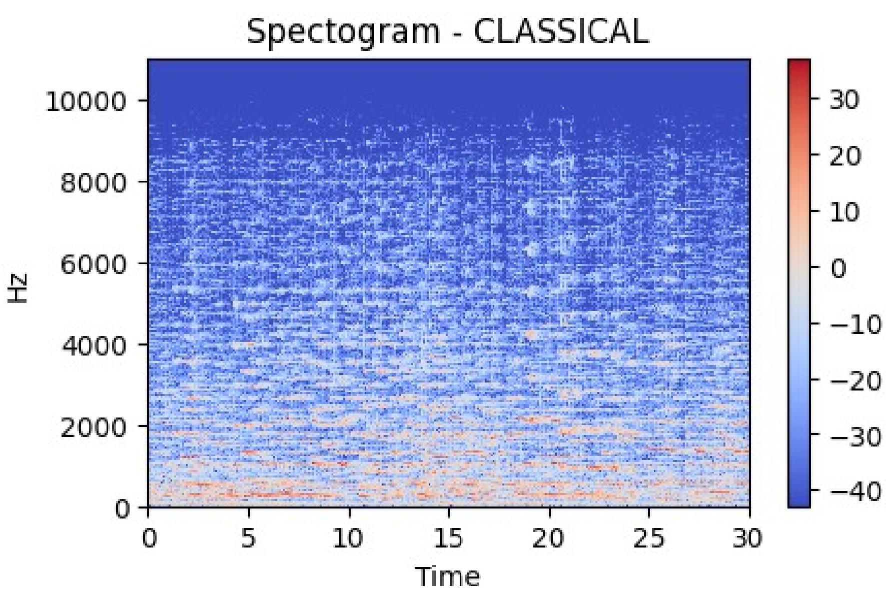



```
# 2. CLASSICAL
audio_classical = r"C:\Users\Shaivya\Desktop\Data\genres_original\
classical\classical.00004.wav"
data_classical, sr = librosa.load(audio_classical)
plt.figure(figsize = (5, 3))
librosa.display.waveshow(data_classical, sr = sr, color = "pink",
alpha = 0.5)
plt.title('Waveplot - CLASSICAL')

# Spectrogram
stft = librosa.stft(data_classical)
stft_db = librosa.amplitude_to_db(abs(stft))
```

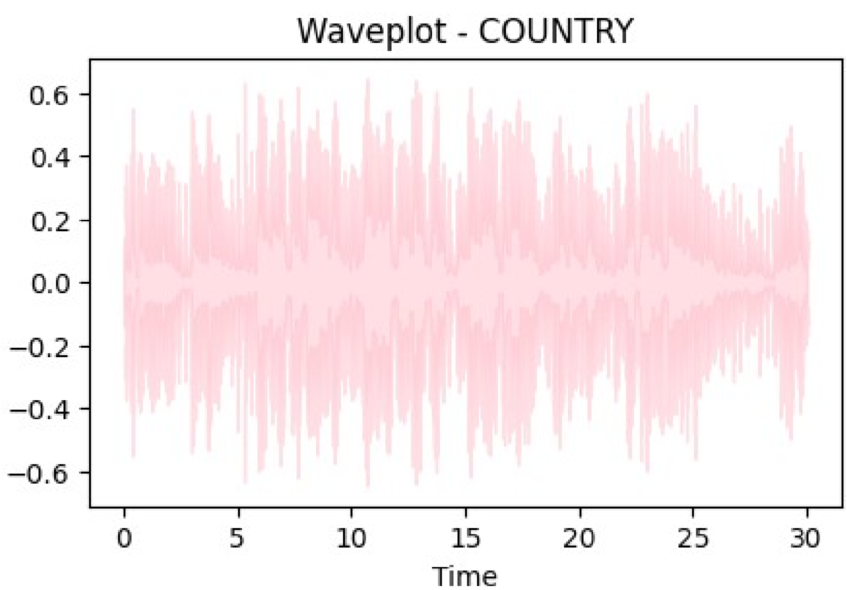
```
plt.figure(figsize = (5, 3))
librosa.display.specshow(stft_db, sr = sr, x_axis = 'time', y_axis =
'hz')
plt.title('Spectogram - CLASSICAL')
plt.colorbar()

# Playing audio
ipd.Audio(audio_classical)
```

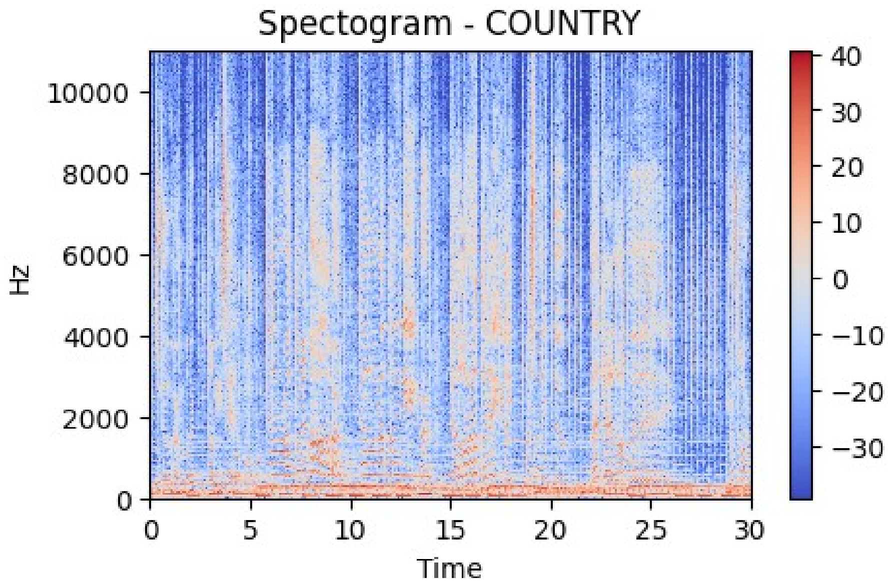
```
<IPython.lib.display.Audio object>
```





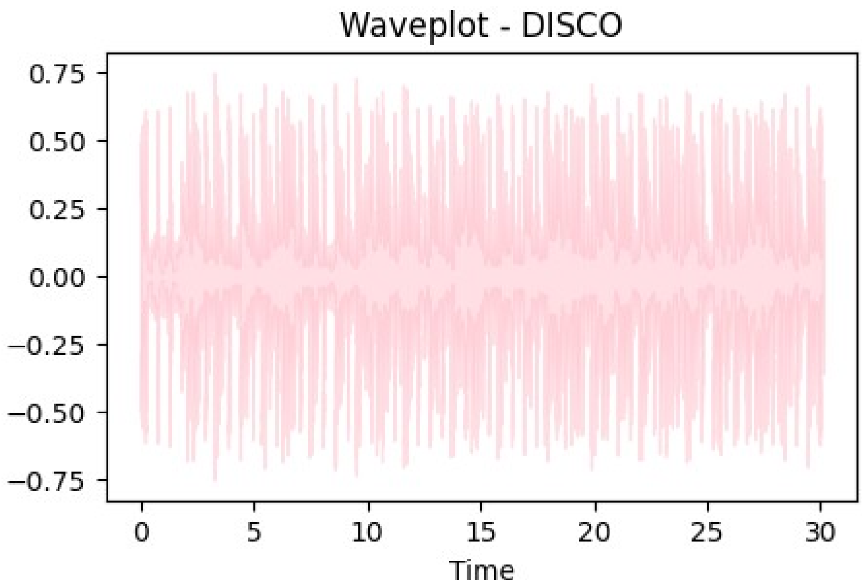
```
# 3. COUNTRY
audio_country = r"C:\Users\Shaivya\Desktop\Data\genres_original\
country\country.00020.wav"
data_country, sr = librosa.load(audio_country)
plt.figure(figsize = (5, 3))
librosa.display.waveshow(data_country, sr = sr, color = "pink", alpha
= 0.5)
plt.title('Waveplot - COUNTRY')

# Spectrogram
stft = librosa.stft(data_country)
stft_db = librosa.amplitude_to_db(abs(stft))
plt.figure(figsize = (5, 3))
librosa.display.specshow(stft_db, sr = sr, x_axis = 'time', y_axis =
'hz')
plt.title('Spectogram - COUNTRY')
plt.colorbar()

# Playing audio
ipd.Audio(audio_country)

<IPython.lib.display.Audio object>
```

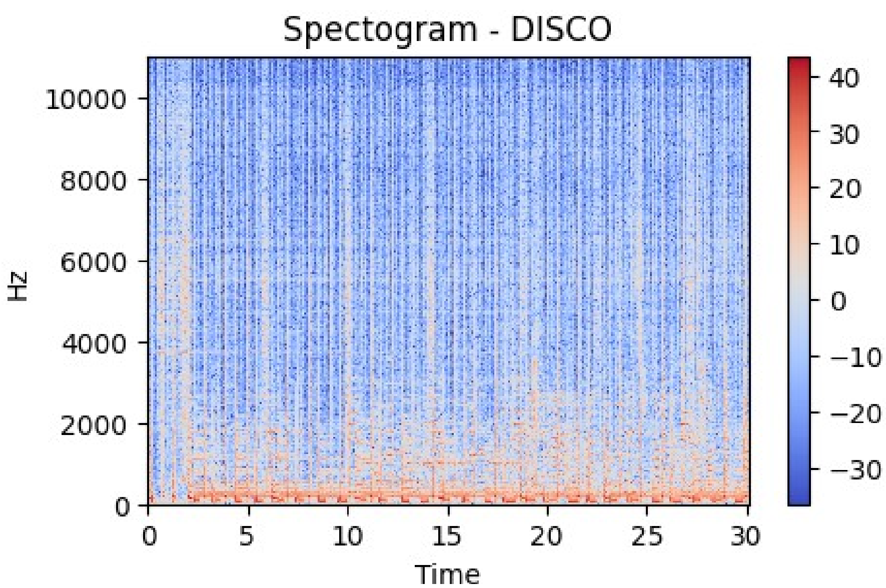





```
# 4. DISCO
audio_disco = r"C:\Users\Shaivya\Desktop\Data\genres_original\disco\
disco.00000.wav"
data_disco, sr = librosa.load(audio_disco)
plt.figure(figsize = (5, 3))
librosa.display.waveshow(data_disco, sr = sr, color = "pink", alpha =
0.5)
plt.title('Waveplot - DISCO')

# Spectrogram
stft = librosa.stft(data_disco)
stft_db = librosa.amplitude_to_db(abs(stft))
plt.figure(figsize = (5, 3))
librosa.display.specshow(stft_db, sr = sr, x_axis = 'time', y_axis =
'hz')
plt.title('Spectogram - DISCO')
plt.colorbar()

# Playing audio
ipd.Audio(audio_disco)
```

```
<IPython.lib.display.Audio object>
```

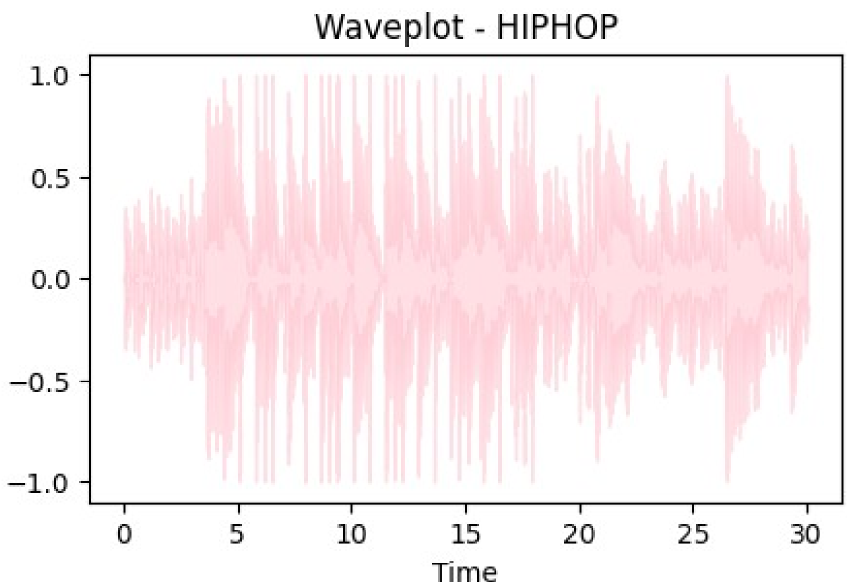


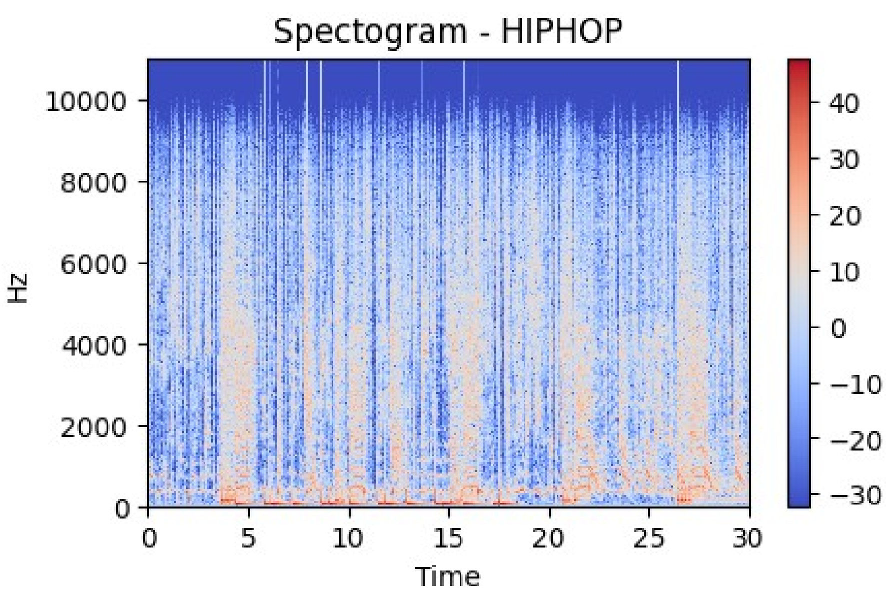



```
# 5. HIPHOP
audio_hiphop = r"C:\Users\Shaivya\Desktop\Data\genres_original\hiphop\
hiphop.00008.wav"
data_hiphop, sr = librosa.load(audio_hiphop)
plt.figure(figsize = (5, 3))
librosa.display.waveshow(data_hiphop, sr = sr, color = "pink", alpha =
0.5)
plt.title('Waveplot - HIPHOP')

# Spectrogram
stft = librosa.stft(data_hiphop)
stft_db = librosa.amplitude_to_db(abs(stft))
```

```
plt.figure(figsize = (5, 3))
librosa.display.specshow(stft_db, sr = sr, x_axis = 'time', y_axis =
'hz')
plt.title('Spectogram - HIPHOP')
plt.colorbar()

# Playing audio
ipd.Audio(audio_hiphop)
```

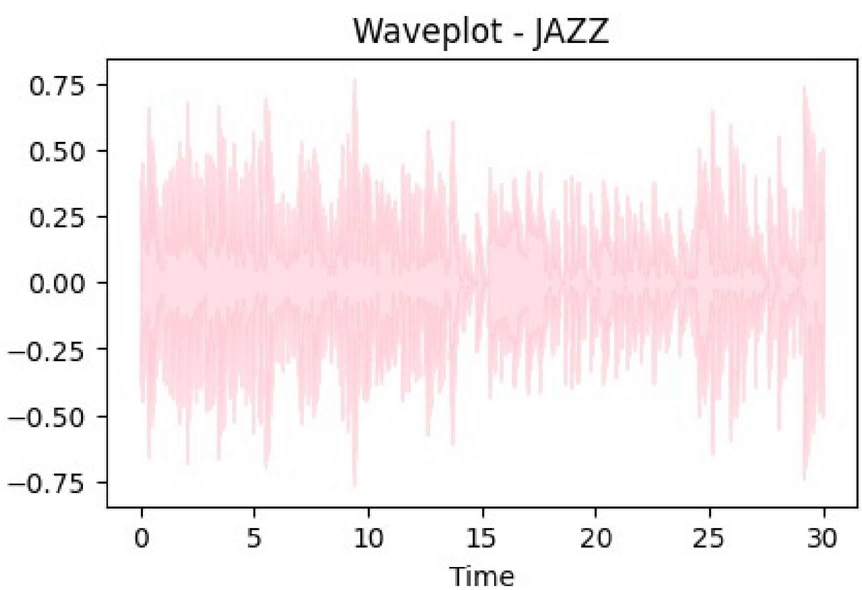
```
<IPython.lib.display.Audio object>
```

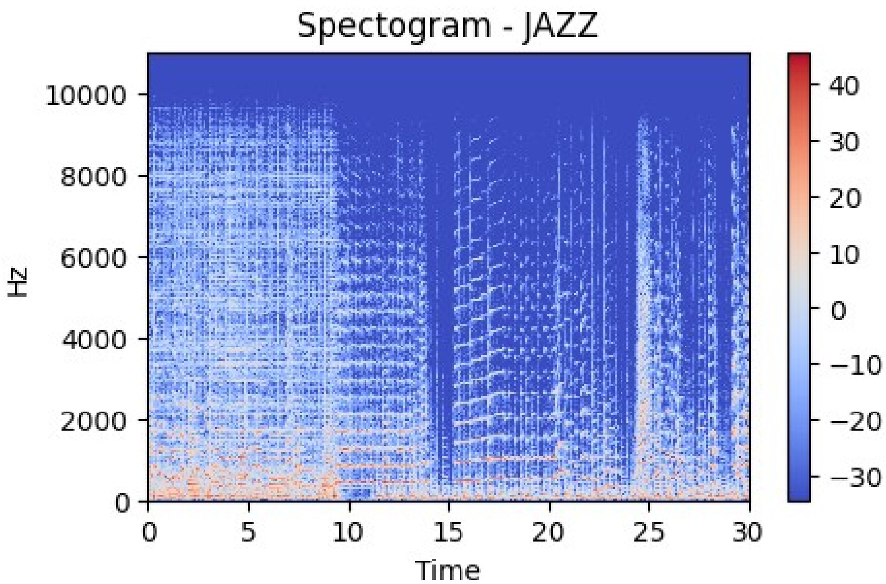




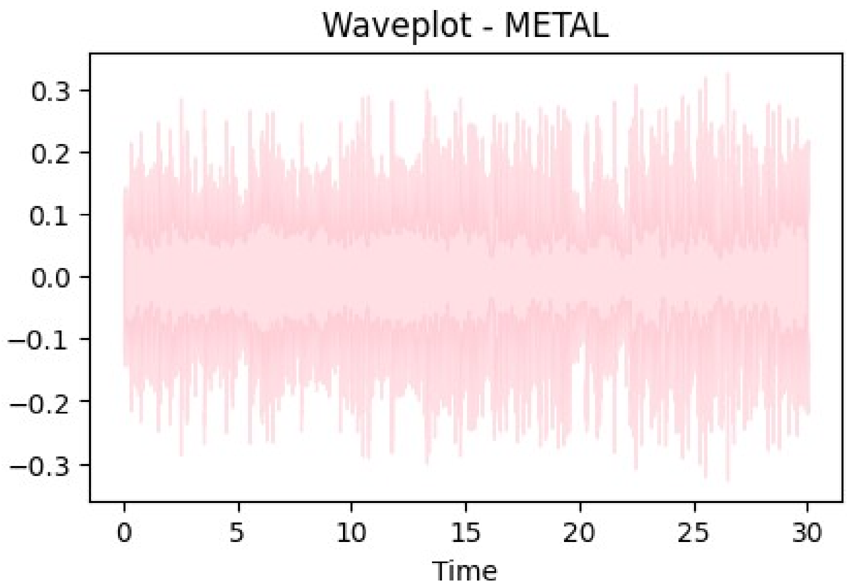
```python
# 6. JAZZ
audio_jazz = r"C:\Users\Shaivya\Desktop\Data\genres_original\jazz\
jazz.00005.wav"
data_jazz, sr = librosa.load(audio_jazz)
plt.figure(figsize = (5, 3))
librosa.display.waveshow(data_jazz, sr = sr, color = "pink", alpha =
0.5)
plt.title('Waveplot - JAZZ')

# Spectrogram
stft = librosa.stft(data_jazz)
stft_db = librosa.amplitude_to_db(abs(stft))
plt.figure(figsize = (5, 3))
librosa.display.specshow(stft_db, sr = sr, x_axis = 'time', y_axis =
'hz')
plt.title('Spectogram - JAZZ')
plt.colorbar()

# Playing audio
ipd.Audio(audio_jazz)
```

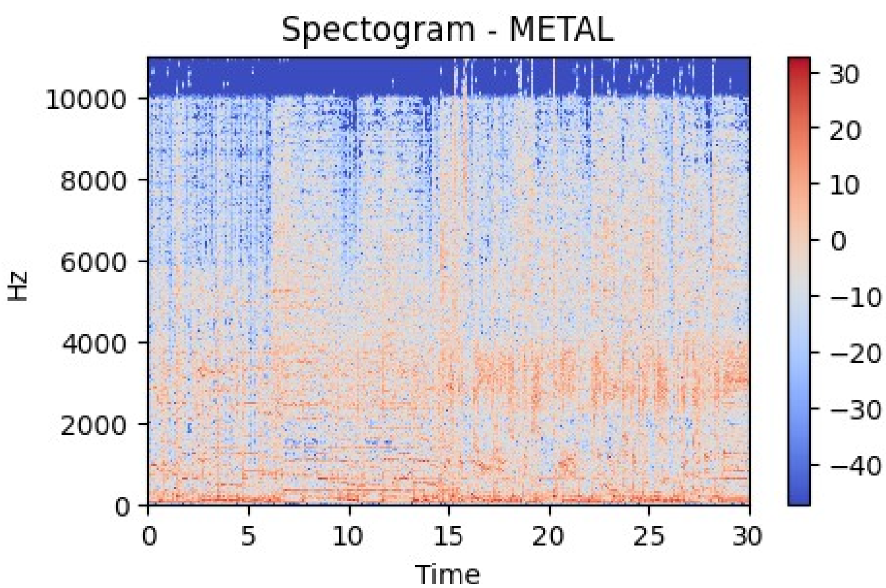
```
<IPython.lib.display.Audio object>
```






```
# 7. METAL
audio_metal = r"C:\Users\Shaivya\Desktop\Data\genres_original\metal\
metal.00006.wav"
data_metal, sr = librosa.load(audio_metal)
plt.figure(figsize = (5, 3))
librosa.display.waveshow(data_metal, sr = sr, color = "pink", alpha =
0.5)
plt.title('Waveplot - METAL')

# Spectrogram
stft = librosa.stft(data_metal)
stft_db = librosa.amplitude_to_db(abs(stft))
plt.figure(figsize = (5, 3))
librosa.display.specshow(stft_db, sr = sr, x_axis = 'time', y_axis =
'hz')
plt.title('Spectogram - METAL')
plt.colorbar()

# Playing audio
ipd.Audio(audio_metal)

<IPython.lib.display.Audio object>
```

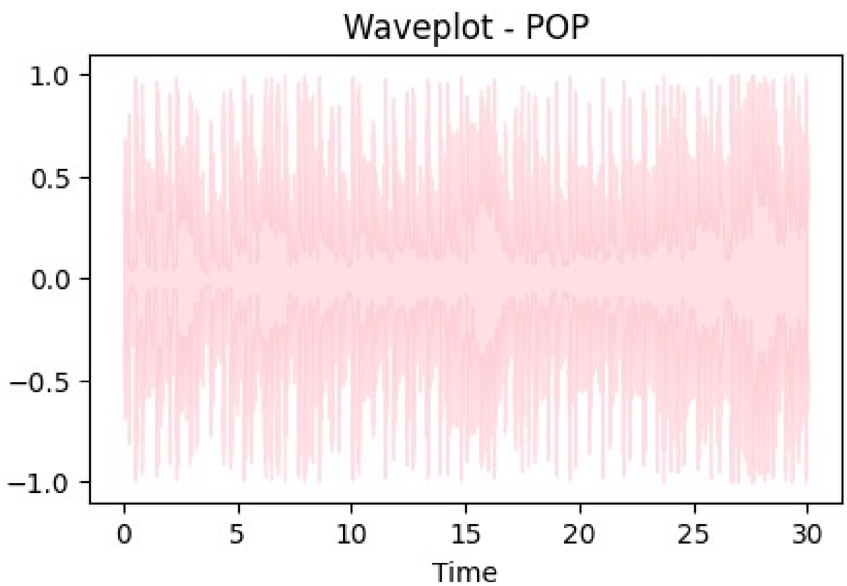


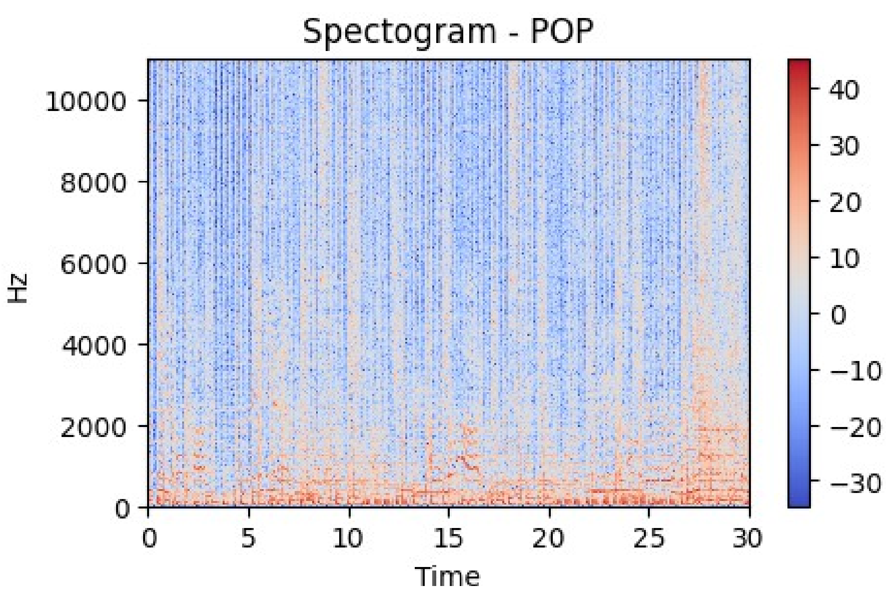



```
# 8. POP
audio_pop = r"C:\Users\Shaivya\Desktop\Data\genres_original\pop\
pop.00028.wav"
data_pop, sr = librosa.load(audio_pop)
plt.figure(figsize = (5, 3))
librosa.display.waveshow(data_pop, sr = sr, color = "pink", alpha =
0.5)
plt.title('Waveplot - POP')

# Spectrogram
stft = librosa.stft(data_pop)
stft_db = librosa.amplitude_to_db(abs(stft))
```

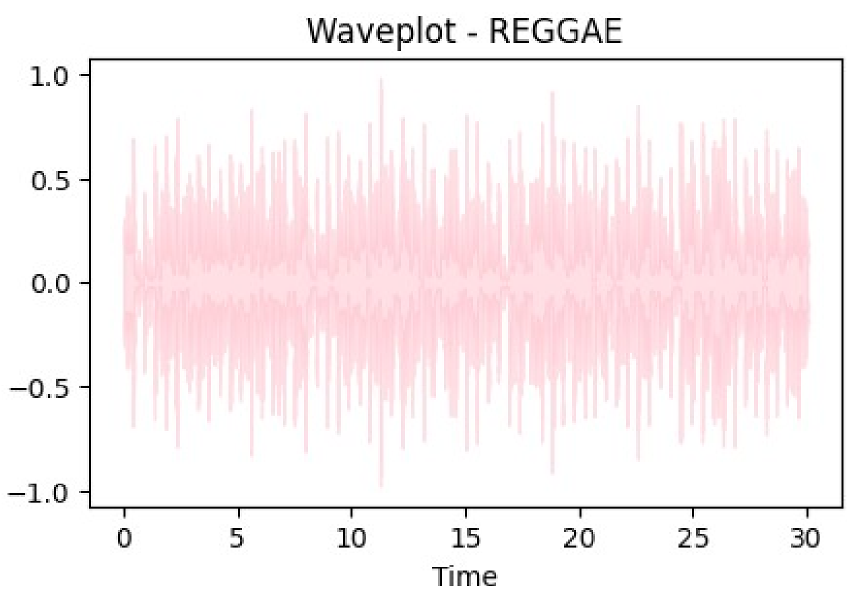
```
plt.figure(figsize = (5, 3))
librosa.display.specshow(stft_db, sr = sr, x_axis = 'time', y_axis =
'hz')
plt.title('Spectogram - POP')
plt.colorbar()

# Playing audio
ipd.Audio(audio_pop)
```

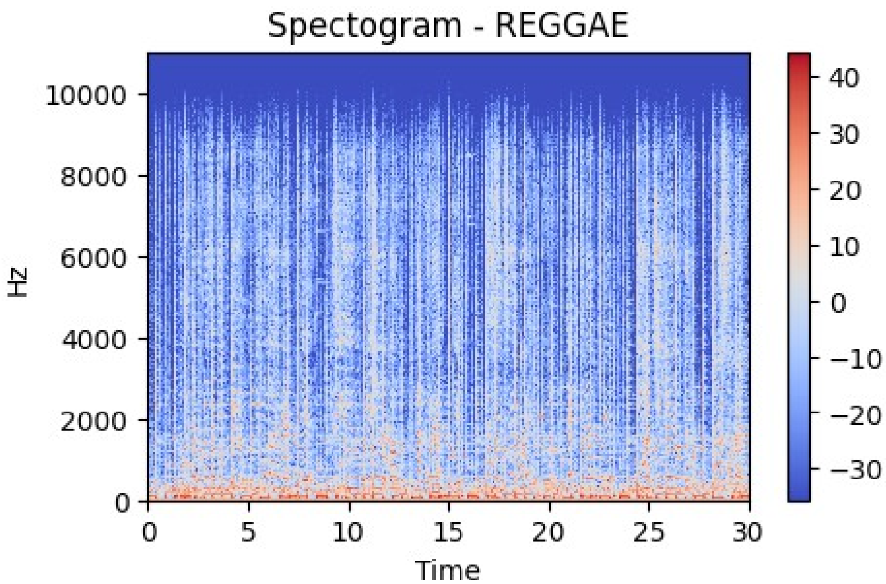
```
<IPython.lib.display.Audio object>
```



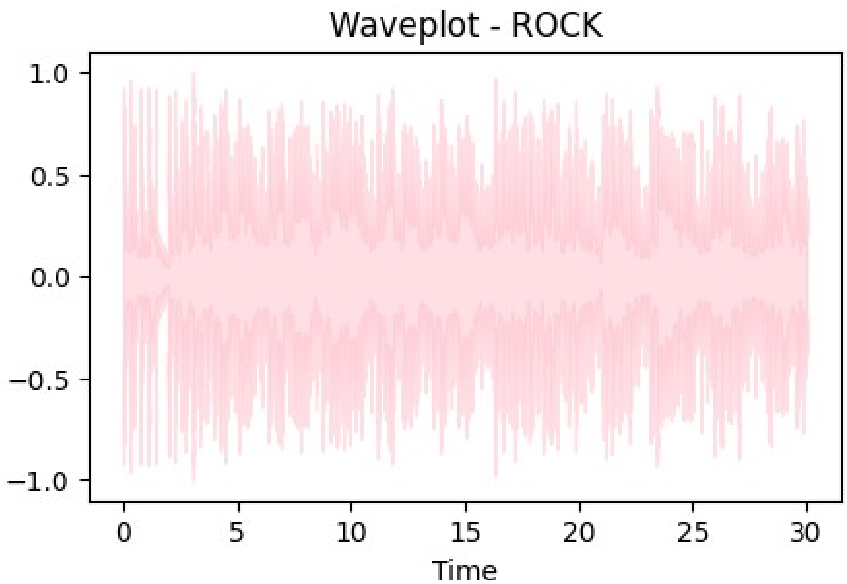


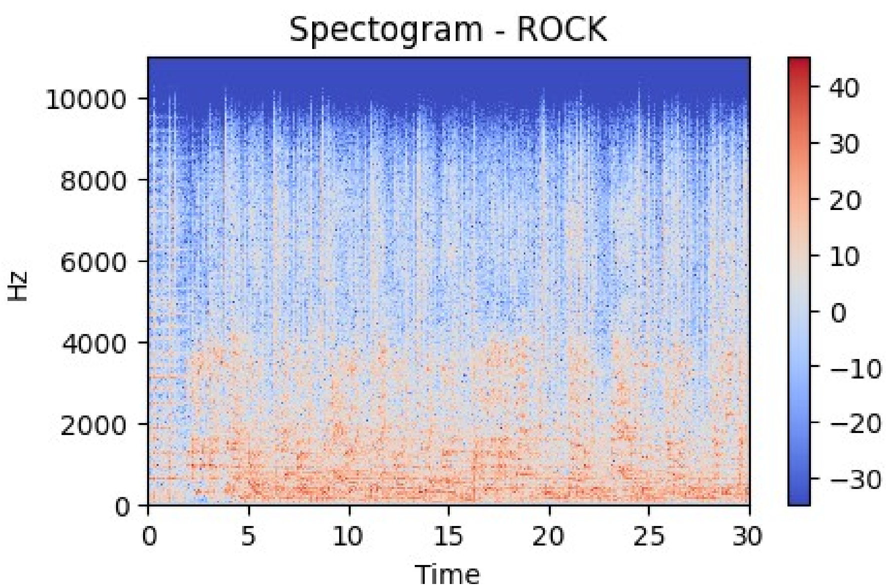
```
# 9. REGGAE
audio_reggae = r"C:\Users\Shaivya\Desktop\Data\genres_original\reggae\
reggae.00020.wav"
data_reggae, sr = librosa.load(audio_reggae)
plt.figure(figsize = (5, 3))
librosa.display.waveshow(data_reggae, sr = sr, color = "pink", alpha =
0.5)
plt.title('Waveplot - REGGAE')

# Spectrogram
stft = librosa.stft(data_reggae)
stft_db = librosa.amplitude_to_db(abs(stft))
plt.figure(figsize = (5, 3))
librosa.display.specshow(stft_db, sr = sr, x_axis = 'time', y_axis =
'hz')
plt.title('Spectogram - REGGAE')
plt.colorbar()

# Playing audio
ipd.Audio(audio_reggae)

<IPython.lib.display.Audio object>
```






```
# 10. ROCK
audio_rock = r"C:\Users\Shaivya\Desktop\Data\genres_original\rock\
rock.00022.wav"
data_rock, sr = librosa.load(audio_rock)
plt.figure(figsize = (5, 3))
librosa.display.waveshow(data_rock, sr = sr, color = "pink", alpha =
0.5)
plt.title('Waveplot - ROCK')

# Spectrogram
stft = librosa.stft(data_rock)
stft_db = librosa.amplitude_to_db(abs(stft))
plt.figure(figsize = (5, 3))
librosa.display.specshow(stft_db, sr = sr, x_axis = 'time', y_axis =
'hz')
plt.title('Spectogram - ROCK')
plt.colorbar()

# Playing audio
ipd.Audio(audio_rock)

<IPython.lib.display.Audio object>
```






```
sound_df.isnull().any()
```

```
filename                      False
length                        False
chroma_stft_mean              False
chroma_stft_var               False
rms_mean                      False
rms_var                       False
spectral_centroid_mean        False
spectral_centroid_var         False
spectral_bandwidth_mean       False
spectral_bandwidth_var        False
```

```
rolloff_mean               False
rolloff_var                False
zero_crossing_rate_mean    False
zero_crossing_rate_var     False
harmony_mean               False
harmony_var                False
perceptr_mean              False
perceptr_var               False
tempo                      False
mfcc1_mean                 False
mfcc1_var                  False
mfcc2_mean                 False
mfcc2_var                  False
mfcc3_mean                 False
mfcc3_var                  False
mfcc4_mean                 False
mfcc4_var                  False
mfcc5_mean                 False
mfcc5_var                  False
mfcc6_mean                 False
mfcc6_var                  False
mfcc7_mean                 False
mfcc7_var                  False
mfcc8_mean                 False
mfcc8_var                  False
mfcc9_mean                 False
mfcc9_var                  False
mfcc10_mean                False
mfcc10_var                 False
mfcc11_mean                False
mfcc11_var                 False
mfcc12_mean                False
mfcc12_var                 False
mfcc13_mean                False
mfcc13_var                 False
mfcc14_mean                False
mfcc14_var                 False
mfcc15_mean                False
mfcc15_var                 False
mfcc16_mean                False
mfcc16_var                 False
mfcc17_mean                False
mfcc17_var                 False
mfcc18_mean                False
mfcc18_var                 False
mfcc19_mean                False
mfcc19_var                 False
mfcc20_mean                False
mfcc20_var                 False
```

```
label                          False
dtype: bool

sound_df = sound_df.drop(labels = "filename", axis = 1)

sound_df

       length  chroma_stft_mean  chroma_stft_var  rms_mean   rms_var  \
0      66149          0.335406         0.091048  0.130405  0.003521
1      66149          0.343065         0.086147  0.112699  0.001450
2      66149          0.346815         0.092243  0.132003  0.004620
3      66149          0.363639         0.086856  0.132565  0.002448
4      66149          0.335579         0.088129  0.143289  0.001701
...      ...               ...              ...       ...       ...
9985   66149          0.349126         0.080515  0.050019  0.000097
9986   66149          0.372564         0.082626  0.057897  0.000088
9987   66149          0.347481         0.089019  0.052403  0.000701
9988   66149          0.387527         0.084815  0.066430  0.000320
9989   66149          0.369293         0.086759  0.050524  0.000067

      spectral_centroid_mean  spectral_centroid_var
spectral_bandwidth_mean  \
0                1773.065032          167541.630869
1972.744388
1                1816.693777           90525.690866
2010.051501
2                1788.539719          111407.437613
2084.565132
3                1655.289045          111952.284517
1960.039988
4                1630.656199           79667.267654
1948.503884
...                      ...                    ...
...
9985             1499.083005          164266.886443
1718.707215
9986             1847.965128          281054.935973
1906.468492
9987             1346.157659          662956.246325
1561.859087
9988             2084.515327          203891.039161
2018.366254
9989             1634.330126          411429.169769
1867.422378

      spectral_bandwidth_var  rolloff_mean  ...  mfcc16_var
mfcc17_mean  \
0              117335.771563   3714.560359  ...   39.687145    -
3.241280
1               65671.875673   3869.682242  ...   64.748276    -
```

```
6.055294
2             75124.921716   3997.639160  ...   67.336563    -
1.768610
3             82913.639269   3568.300218  ...   47.739452    -
3.841155
4             60204.020268   3469.992864  ...   30.336359
0.664582
...                    ...           ...  ...         ...          .
..
9985          85931.574523   3015.559458  ...   42.485981    -
9.094270
9986          99727.037054   3746.694524  ...   32.415203   -
12.375726
9987         138762.841945   2442.362154  ...   78.228149    -
2.524483
9988          22860.992562   4313.266226  ...   28.323744    -
5.363541
9989         119722.211518   3462.042142  ...   38.801735   -
11.598399

      mfcc17_var  mfcc18_mean  mfcc18_var  mfcc19_mean  mfcc19_var  \
0      36.488243     0.722209   38.099152    -5.050335   33.618073
1      40.677654     0.159015   51.264091    -2.837699   97.030830
2      28.348579     2.378768   45.717648    -1.938424   53.050835
3      28.337118     1.218588   34.770935    -3.580352   50.836224
4      45.880913     1.689446   51.363583    -3.392489   26.738789
...          ...          ...         ...          ...         ...
9985   38.326839    -4.246976   31.049839    -5.625813   48.804092
9986   66.418587    -3.081278   54.414265   -11.960546   63.452255
9987   21.778994     4.809936   25.980829     1.775686   48.582378
9988   17.209942     6.462601   21.442928     2.354765   24.843613
9989   58.983097    -0.178517   55.761299    -6.903252   39.485901

      mfcc20_mean  mfcc20_var  label
0       -0.243027   43.771767  blues
1        5.784063   59.943081  blues
2        2.517375   33.105122  blues
3        3.630866   32.023678  blues
4        0.536961   29.146694  blues
...           ...         ...    ...
9985     1.818823   38.966969   rock
9986     0.428857   18.697033   rock
9987    -0.299545   41.586990   rock
9988     0.675824   12.787750   rock
9989    -3.412534   31.727489   rock

[9990 rows x 59 columns]

from sklearn.preprocessing import LabelEncoder
import pandas as pd
```

```
class_list = sound_df.iloc[:, -1]
convertor = LabelEncoder()
y = convertor.fit_transform(class_list)
y
```

```
array([0, 0, 0, ..., 9, 9, 9])
```

```
print(sound_df.iloc[:, :-1])
```

```
      length  chroma_stft_mean  chroma_stft_var  rms_mean   rms_var  \
0      66149          0.335406         0.091048  0.130405  0.003521
1      66149          0.343065         0.086147  0.112699  0.001450
2      66149          0.346815         0.092243  0.132003  0.004620
3      66149          0.363639         0.086856  0.132565  0.002448
4      66149          0.335579         0.088129  0.143289  0.001701
...      ...               ...              ...       ...       ...
9985   66149          0.349126         0.080515  0.050019  0.000097
9986   66149          0.372564         0.082626  0.057897  0.000088
9987   66149          0.347481         0.089019  0.052403  0.000701
9988   66149          0.387527         0.084815  0.066430  0.000320
9989   66149          0.369293         0.086759  0.050524  0.000067

      spectral_centroid_mean  spectral_centroid_var
spectral_bandwidth_mean  \
0                1773.065032          167541.630869
1972.744388
1                1816.693777           90525.690866
2010.051501
2                1788.539719          111407.437613
2084.565132
3                1655.289045          111952.284517
1960.039988
4                1630.656199           79667.267654
1948.503884
...                      ...                    ...
...
9985             1499.083005          164266.886443
1718.707215
9986             1847.965128          281054.935973
1906.468492
9987             1346.157659          662956.246325
1561.859087
9988             2084.515327          203891.039161
2018.366254
9989             1634.330126          411429.169769
1867.422378

      spectral_bandwidth_var  rolloff_mean  ...  mfcc16_mean
mfcc16_var  \
```

```
0               117335.771563   3714.560359  ...    -2.853603
39.687145
1                65671.875673   3869.682242  ...     4.074709
64.748276
2                75124.921716   3997.639160  ...     4.806280
67.336563
3                82913.639269   3568.300218  ...    -1.359111
47.739452
4                60204.020268   3469.992864  ...     2.092937
30.336359
...                       ...           ...  ...          ...          .
..
9985             85931.574523   3015.559458  ...     5.773784
42.485981
9986             99727.037054   3746.694524  ...     2.074155
32.415203
9987            138762.841945   2442.362154  ...    -1.005473
78.228149
9988             22860.992562   4313.266226  ...     4.123402
28.323744
9989            119722.211518   3462.042142  ...     1.342274
38.801735

      mfcc17_mean  mfcc17_var  mfcc18_mean  mfcc18_var  mfcc19_mean  \
0       -3.241280   36.488243     0.722209   38.099152    -5.050335
1       -6.055294   40.677654     0.159015   51.264091    -2.837699
2       -1.768610   28.348579     2.378768   45.717648    -1.938424
3       -3.841155   28.337118     1.218588   34.770935    -3.580352
4        0.664582   45.880913     1.689446   51.363583    -3.392489
...           ...         ...          ...         ...          ...
9985    -9.094270   38.326839    -4.246976   31.049839    -5.625813
9986   -12.375726   66.418587    -3.081278   54.414265   -11.960546
9987    -2.524483   21.778994     4.809936   25.980829     1.775686
9988    -5.363541   17.209942     6.462601   21.442928     2.354765
9989   -11.598399   58.983097    -0.178517   55.761299    -6.903252

      mfcc19_var  mfcc20_mean  mfcc20_var
0      33.618073    -0.243027   43.771767
1      97.030830     5.784063   59.943081
2      53.050835     2.517375   33.105122
3      50.836224     3.630866   32.023678
4      26.738789     0.536961   29.146694
...          ...          ...         ...
9985   48.804092     1.818823   38.966969
9986   63.452255     0.428857   18.697033
9987   48.582378    -0.299545   41.586990
9988   24.843613     0.675824   12.787750
9989   39.485901    -3.412534   31.727489

[9990 rows x 58 columns]
```

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

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size =
0.33)

len(y_train)

6693

len(y_test)

3297

x_train.shape, x_test.shape

((6693, 58), (3297, 58))

y_train.shape, y_test.shape

((6693,), (3297,))

import os
import math
import json
import librosa
import warnings
import soundfile as sf

warnings.filterwarnings("ignore", category = UserWarning, message =
"PySoundFile failed. Trying audioread instead.")
warnings.filterwarnings("ignore", category = FutureWarning, message =
"librosa.core.audio.__audioread_load.*")

DATASET_PATH = r"C:\Users\Shaivya\Desktop\Data\genres_original"
JSON_PATH = "data_10.json"
SAMPLE_RATE = 22050
TRACK_DURATION = 30
SAMPLES_PER_TRACK = SAMPLE_RATE * TRACK_DURATION

def save_mfcc(dataset_path, json_path, num_mfcc=13, n_fft=2048,
hop_length=512, num_segments=5):
    data = {
        "mapping": [],
        "labels": [],
        "mfcc": []
    }

    samples_per_segment = int(SAMPLES_PER_TRACK / num_segments)
    num_mfcc_vectors_per_segment = math.ceil(samples_per_segment /
hop_length)
```

```
    for i, (dirpath, dirnames, filenames) in
enumerate(os.walk(dataset_path)):
        if dirpath != dataset_path:
            semantic_label = dirpath.split("/")[-1]
            data["mapping"].append(semantic_label)
            print(semantic_label)
            for f in filenames:
                file_path = os.path.join(dirpath, f)

                # Check if the file format is supported
                try:
                    sf.info(file_path)
                except:
                    print(f"Skipping file {file_path} due to
unsupported format.")
                    continue

                signal, sample_rate = librosa.load(file_path,
sr=SAMPLE_RATE)
                for d in range(num_segments):
                    start = samples_per_segment * d
                    finish = start + samples_per_segment
                    mfcc = librosa.feature.mfcc(y =
signal[start:finish], sr = sample_rate, n_mfcc = num_mfcc, n_fft =
n_fft, hop_length = hop_length)
                    mfcc = mfcc.T
                    if len(mfcc) == num_mfcc_vectors_per_segment:
                        data["mfcc"].append(mfcc.tolist())
                        data["labels"].append(i-1)

    with open(json_path, "w") as fp:
        json.dump(data, fp, indent = 4)

save_mfcc(DATASET_PATH, JSON_PATH, num_segments = 15)
```

```
C:\Users\Shaivya\Desktop\Data\genres_original\blues
C:\Users\Shaivya\Desktop\Data\genres_original\classical
C:\Users\Shaivya\Desktop\Data\genres_original\country
C:\Users\Shaivya\Desktop\Data\genres_original\disco
C:\Users\Shaivya\Desktop\Data\genres_original\hiphop
C:\Users\Shaivya\Desktop\Data\genres_original\jazz
Skipping file C:\Users\Shaivya\Desktop\Data\genres_original\jazz\
jazz.00054.wav due to unsupported format.
C:\Users\Shaivya\Desktop\Data\genres_original\metal
C:\Users\Shaivya\Desktop\Data\genres_original\pop
C:\Users\Shaivya\Desktop\Data\genres_original\reggae
C:\Users\Shaivya\Desktop\Data\genres_original\rock
```

```
def prepare_datasets(test_size, validation_size):
    x, y, z = load_data(DATA_PATH)
    x_train, x_test, y_train, y_test = train_test_split(x, y,
test_size = test_size, shuffle = True,random_state =42)
    x_train, x_validation, y_train, y_validation =
train_test_split(x_train, y_train, test_size=1, shuffle = True,
random_state = 42)

    # add an axis to input sets
    x_train = x_train[..., np.newaxis]
    x_validation = x_validation[..., np.newaxis]
    x_test = x_test[..., np.newaxis]

    return x_train, x_validation, x_test, y_train, y_validation,
y_test, z

x_train.shape, x_test.shape

((6693, 58), (3297, 58))

DATA_PATH = "./data_10.json"
def load_data(data_path):

    with open(data_path, "r") as fp:
        data = json.load(fp)

    x = np.array(data["mfcc"])
    y = np.array(data["labels"])
    z = np.array(data['mapping'])
    return x, y, z

from keras.models import Sequential
from keras.layers import *

def build_model(input_shape):
    model = Sequential()

    #1st conv layer
    model.add(Conv2D(32, (2, 2), activation='relu',
input_shape=input_shape, kernel_initializer = 'he_normal'))
    model.add(MaxPooling2D((3, 3), strides=(2, 2), padding='same'))
    model.add(BatchNormalization())

    # 2nd conv layer
    model.add(Conv2D(32, (2, 2), activation='relu', kernel_initializer
= 'he_normal'))
    model.add(MaxPooling2D((3, 3), strides=(2, 2), padding='same'))
    model.add(BatchNormalization())

    # 3rd conv layer
    model.add(Conv2D(32, (2, 2), activation='relu', kernel_initializer
```

```
= 'he_normal'))
    model.add(MaxPooling2D((2, 2), strides=(2, 2), padding='same'))
    model.add(BatchNormalization())

    # flatten output and feed it into dense layer
    model.add(Flatten())
    model.add(Dense(128, activation='relu', kernel_initializer =
'he_normal'))
    model.add(Dropout(0.5))
    model.add(Dense(64, activation='relu', kernel_initializer =
'he_normal'))
    model.add(Dropout(0.5))

    # output layer
    model.add(Dense(10, activation='softmax'))
    return model

x_train, x_validation, x_test, y_train, y_validation, y_test, z =
prepare_datasets(0.1, 0)

input_shape = (x_train.shape[1], x_train.shape[2], 1)
model = build_model(input_shape)
model.summary()
```

```
C:\Users\Shaivya\AppData\Roaming\Python\Python311\site-packages\keras\
src\layers\convolutional\base_conv.py:107: UserWarning: Do not pass an
`input_shape`/`input_dim` argument to a layer. When using Sequential
models, prefer using an `Input(shape)` object as the first layer in
the model instead.
  super().__init__(activity_regularizer=activity_regularizer,
**kwargs)
```

```
Model: "sequential_1"
```

```
┏━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┓
━━━━━━━━━━━━━━━━━┓
┃ Layer (type)                          ┃ Output Shape                 ┃
Param # ┃
┡━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┩
━━━━━━━━━━━━━━━━━┩
│ conv2d_3 (Conv2D)                     │ (None, 86, 12, 32)           │
160 │
├───────────────────────────────────────┼──────────────────────────────┤
─────────────────┤
│ max_pooling2d_3 (MaxPooling2D)        │ (None, 43, 6, 32)            │
0 │
├───────────────────────────────────────┼──────────────────────────────┤
─────────────────┤
│ batch_normalization_3                 │ (None, 43, 6, 32)            │
128 │
```

```
│ (BatchNormalization)                   │                              │
│
├────────────────────────────────────────┼──────────────────────────────┤
─────────────────┤
│ conv2d_4 (Conv2D)                      │ (None, 42, 5, 32)            │
4,128 │
├────────────────────────────────────────┼──────────────────────────────┤
─────────────────┤
│ max_pooling2d_4 (MaxPooling2D)         │ (None, 21, 3, 32)            │
0 │
├────────────────────────────────────────┼──────────────────────────────┤
─────────────────┤
│ batch_normalization_4                  │ (None, 21, 3, 32)            │
128 │
│ (BatchNormalization)                   │                              │
│
├────────────────────────────────────────┼──────────────────────────────┤
─────────────────┤
│ conv2d_5 (Conv2D)                      │ (None, 20, 2, 32)            │
4,128 │
├────────────────────────────────────────┼──────────────────────────────┤
─────────────────┤
│ max_pooling2d_5 (MaxPooling2D)         │ (None, 10, 1, 32)            │
0 │
├────────────────────────────────────────┼──────────────────────────────┤
─────────────────┤
│ batch_normalization_5                  │ (None, 10, 1, 32)            │
128 │
│ (BatchNormalization)                   │                              │
│
├────────────────────────────────────────┼──────────────────────────────┤
─────────────────┤
│ flatten_1 (Flatten)                    │ (None, 320)                  │
0 │
├────────────────────────────────────────┼──────────────────────────────┤
─────────────────┤
│ dense_3 (Dense)                        │ (None, 128)                  │
41,088 │
├────────────────────────────────────────┼──────────────────────────────┤
─────────────────┤
│ dropout_2 (Dropout)                    │ (None, 128)                  │
0 │
├────────────────────────────────────────┼──────────────────────────────┤
─────────────────┤
│ dense_4 (Dense)                        │ (None, 64)                   │
8,256 │
├────────────────────────────────────────┼──────────────────────────────┤
─────────────────┤
│ dropout_3 (Dropout)                    │ (None, 64)                   │
```

```
0 │
├─────────────────────────────────────┼─────────────────────────────┼
─────────────────┤
│ dense_5 (Dense)                     │ (None, 10)                  │
650 │
└─────────────────────────────────────┴─────────────────────────────┴
─────────────────┘

 Total params: 58,794 (229.66 KB)

 Trainable params: 58,602 (228.91 KB)

 Non-trainable params: 192 (768.00 B)
```

```
def predict(model, x, y):

    x = x[np.newaxis, ...]
    prediction = model.predict(x)
    predicted_index = np.argmax(prediction, axis = 1)
    target = z[y]
    predicted = z[predicted_index]

    print("\nActual Label: {}\nPredicted Label: {}".format(target,
predicted))

import tensorflow as tf
from tensorflow.keras import optimizers

optimiser = optimizers.Adam(learning_rate=0.0025)
model.compile(optimizer = optimiser, loss =
'sparse_categorical_crossentropy', metrics = ['accuracy'])

history = model.fit(x_train, y_train, validation_data = (x_test,
y_test), batch_size = 512, epochs = 150, verbose = 2)
```

```
Epoch 1/150
27/27 - 8s - 290ms/step - accuracy: 0.2233 - loss: 2.5655 -
val_accuracy: 0.1749 - val_loss: 3.1780
Epoch 2/150
27/27 - 4s - 152ms/step - accuracy: 0.3384 - loss: 1.8316 -
val_accuracy: 0.2089 - val_loss: 2.5436
Epoch 3/150
27/27 - 4s - 156ms/step - accuracy: 0.3901 - loss: 1.6676 -
val_accuracy: 0.2183 - val_loss: 2.6495
Epoch 4/150
27/27 - 4s - 151ms/step - accuracy: 0.4361 - loss: 1.5461 -
val_accuracy: 0.2904 - val_loss: 1.9320
Epoch 5/150
27/27 - 4s - 153ms/step - accuracy: 0.4633 - loss: 1.4497 -
val_accuracy: 0.3732 - val_loss: 1.6886
Epoch 6/150
```

```
27/27 - 4s - 148ms/step - accuracy: 0.4882 - loss: 1.3845 -
val_accuracy: 0.4312 - val_loss: 1.5307
Epoch 7/150
27/27 - 4s - 150ms/step - accuracy: 0.5164 - loss: 1.3166 -
val_accuracy: 0.4913 - val_loss: 1.3810
Epoch 8/150
27/27 - 4s - 150ms/step - accuracy: 0.5319 - loss: 1.2750 -
val_accuracy: 0.5507 - val_loss: 1.2204
Epoch 9/150
27/27 - 4s - 149ms/step - accuracy: 0.5509 - loss: 1.2312 -
val_accuracy: 0.4980 - val_loss: 1.3331
Epoch 10/150
27/27 - 4s - 158ms/step - accuracy: 0.5659 - loss: 1.1894 -
val_accuracy: 0.5788 - val_loss: 1.1788
Epoch 11/150
27/27 - 4s - 153ms/step - accuracy: 0.5868 - loss: 1.1590 -
val_accuracy: 0.6128 - val_loss: 1.0623
Epoch 12/150
27/27 - 4s - 157ms/step - accuracy: 0.5983 - loss: 1.1201 -
val_accuracy: 0.6162 - val_loss: 1.0640
Epoch 13/150
27/27 - 4s - 162ms/step - accuracy: 0.6119 - loss: 1.0800 -
val_accuracy: 0.6288 - val_loss: 1.0172
Epoch 14/150
27/27 - 4s - 153ms/step - accuracy: 0.6255 - loss: 1.0591 -
val_accuracy: 0.6442 - val_loss: 0.9832
Epoch 15/150
27/27 - 5s - 178ms/step - accuracy: 0.6358 - loss: 1.0329 -
val_accuracy: 0.6676 - val_loss: 0.9401
Epoch 16/150
27/27 - 4s - 150ms/step - accuracy: 0.6482 - loss: 1.0049 -
val_accuracy: 0.5901 - val_loss: 1.1286
Epoch 17/150
27/27 - 4s - 158ms/step - accuracy: 0.6649 - loss: 0.9647 -
val_accuracy: 0.5474 - val_loss: 1.4583
Epoch 18/150
27/27 - 5s - 173ms/step - accuracy: 0.6753 - loss: 0.9295 -
val_accuracy: 0.6248 - val_loss: 1.0995
Epoch 19/150
27/27 - 4s - 166ms/step - accuracy: 0.6875 - loss: 0.9177 -
val_accuracy: 0.6756 - val_loss: 0.9387
Epoch 20/150
27/27 - 4s - 156ms/step - accuracy: 0.6895 - loss: 0.8987 -
val_accuracy: 0.6629 - val_loss: 0.9963
Epoch 21/150
27/27 - 4s - 158ms/step - accuracy: 0.7013 - loss: 0.8665 -
val_accuracy: 0.6709 - val_loss: 0.9483
Epoch 22/150
27/27 - 4s - 153ms/step - accuracy: 0.7099 - loss: 0.8389 -
```

```
val_accuracy: 0.6936 - val_loss: 0.8692
Epoch 23/150
27/27 - 4s - 153ms/step - accuracy: 0.7200 - loss: 0.8142 -
val_accuracy: 0.7190 - val_loss: 0.8299
Epoch 24/150
27/27 - 4s - 153ms/step - accuracy: 0.7256 - loss: 0.7967 -
val_accuracy: 0.7203 - val_loss: 0.7915
Epoch 25/150
27/27 - 4s - 160ms/step - accuracy: 0.7328 - loss: 0.7817 -
val_accuracy: 0.6736 - val_loss: 1.0392
Epoch 26/150
27/27 - 4s - 152ms/step - accuracy: 0.7408 - loss: 0.7675 -
val_accuracy: 0.7150 - val_loss: 0.8647
Epoch 27/150
27/27 - 4s - 157ms/step - accuracy: 0.7450 - loss: 0.7447 -
val_accuracy: 0.7196 - val_loss: 0.8279
Epoch 28/150
27/27 - 4s - 150ms/step - accuracy: 0.7485 - loss: 0.7337 -
val_accuracy: 0.7483 - val_loss: 0.7515
Epoch 29/150
27/27 - 4s - 164ms/step - accuracy: 0.7589 - loss: 0.7131 -
val_accuracy: 0.7303 - val_loss: 0.8211
Epoch 30/150
27/27 - 4s - 157ms/step - accuracy: 0.7662 - loss: 0.6899 -
val_accuracy: 0.7183 - val_loss: 0.8895
Epoch 31/150
27/27 - 4s - 155ms/step - accuracy: 0.7674 - loss: 0.6939 -
val_accuracy: 0.7196 - val_loss: 0.8321
Epoch 32/150
27/27 - 4s - 159ms/step - accuracy: 0.7687 - loss: 0.6790 -
val_accuracy: 0.7457 - val_loss: 0.7590
Epoch 33/150
27/27 - 4s - 149ms/step - accuracy: 0.7865 - loss: 0.6354 -
val_accuracy: 0.7437 - val_loss: 0.7931
Epoch 34/150
27/27 - 4s - 159ms/step - accuracy: 0.7747 - loss: 0.6653 -
val_accuracy: 0.7190 - val_loss: 0.8222
Epoch 35/150
27/27 - 4s - 156ms/step - accuracy: 0.7832 - loss: 0.6351 -
val_accuracy: 0.7750 - val_loss: 0.7033
Epoch 36/150
27/27 - 4s - 160ms/step - accuracy: 0.7962 - loss: 0.6026 -
val_accuracy: 0.7517 - val_loss: 0.7436
Epoch 37/150
27/27 - 4s - 157ms/step - accuracy: 0.8012 - loss: 0.5939 -
val_accuracy: 0.7810 - val_loss: 0.7195
Epoch 38/150
27/27 - 5s - 167ms/step - accuracy: 0.8003 - loss: 0.5909 -
val_accuracy: 0.7570 - val_loss: 0.7690
```

```
Epoch 39/150
27/27 - 5s - 185ms/step - accuracy: 0.8045 - loss: 0.5882 -
val_accuracy: 0.7563 - val_loss: 0.7242
Epoch 40/150
27/27 - 4s - 165ms/step - accuracy: 0.8067 - loss: 0.5776 -
val_accuracy: 0.7677 - val_loss: 0.7300
Epoch 41/150
27/27 - 4s - 165ms/step - accuracy: 0.8068 - loss: 0.5683 -
val_accuracy: 0.7550 - val_loss: 0.7856
Epoch 42/150
27/27 - 5s - 168ms/step - accuracy: 0.8134 - loss: 0.5524 -
val_accuracy: 0.7690 - val_loss: 0.7449
Epoch 43/150
27/27 - 5s - 179ms/step - accuracy: 0.8173 - loss: 0.5489 -
val_accuracy: 0.7637 - val_loss: 0.7369
Epoch 44/150
27/27 - 5s - 186ms/step - accuracy: 0.8201 - loss: 0.5406 -
val_accuracy: 0.7563 - val_loss: 0.8353
Epoch 45/150
27/27 - 5s - 189ms/step - accuracy: 0.8208 - loss: 0.5236 -
val_accuracy: 0.7523 - val_loss: 0.8222
Epoch 46/150
27/27 - 5s - 173ms/step - accuracy: 0.8233 - loss: 0.5169 -
val_accuracy: 0.7617 - val_loss: 0.7797
Epoch 47/150
27/27 - 4s - 155ms/step - accuracy: 0.8341 - loss: 0.4960 -
val_accuracy: 0.7857 - val_loss: 0.7235
Epoch 48/150
27/27 - 4s - 149ms/step - accuracy: 0.8322 - loss: 0.5055 -
val_accuracy: 0.7503 - val_loss: 0.7968
Epoch 49/150
27/27 - 4s - 150ms/step - accuracy: 0.8365 - loss: 0.4873 -
val_accuracy: 0.7704 - val_loss: 0.7823
Epoch 50/150
27/27 - 4s - 147ms/step - accuracy: 0.8353 - loss: 0.4908 -
val_accuracy: 0.7797 - val_loss: 0.7389
Epoch 51/150
27/27 - 4s - 147ms/step - accuracy: 0.8388 - loss: 0.4832 -
val_accuracy: 0.7757 - val_loss: 0.7346
Epoch 52/150
27/27 - 4s - 153ms/step - accuracy: 0.8382 - loss: 0.4824 -
val_accuracy: 0.7864 - val_loss: 0.7257
Epoch 53/150
27/27 - 4s - 148ms/step - accuracy: 0.8492 - loss: 0.4488 -
val_accuracy: 0.7864 - val_loss: 0.7141
Epoch 54/150
27/27 - 4s - 151ms/step - accuracy: 0.8507 - loss: 0.4474 -
val_accuracy: 0.7724 - val_loss: 0.7965
Epoch 55/150
```

```
27/27 - 4s - 153ms/step - accuracy: 0.8472 - loss: 0.4567 -
val_accuracy: 0.7637 - val_loss: 0.8239
Epoch 56/150
27/27 - 4s - 148ms/step - accuracy: 0.8550 - loss: 0.4494 -
val_accuracy: 0.7837 - val_loss: 0.6994
Epoch 57/150
27/27 - 4s - 149ms/step - accuracy: 0.8538 - loss: 0.4321 -
val_accuracy: 0.7617 - val_loss: 0.8498
Epoch 58/150
27/27 - 4s - 159ms/step - accuracy: 0.8560 - loss: 0.4317 -
val_accuracy: 0.7784 - val_loss: 0.7610
Epoch 59/150
27/27 - 5s - 167ms/step - accuracy: 0.8563 - loss: 0.4250 -
val_accuracy: 0.7837 - val_loss: 0.7426
Epoch 60/150
27/27 - 5s - 181ms/step - accuracy: 0.8618 - loss: 0.4162 -
val_accuracy: 0.7710 - val_loss: 0.8507
Epoch 61/150
27/27 - 5s - 186ms/step - accuracy: 0.8627 - loss: 0.4023 -
val_accuracy: 0.7770 - val_loss: 0.8111
Epoch 62/150
27/27 - 5s - 174ms/step - accuracy: 0.8635 - loss: 0.4152 -
val_accuracy: 0.7824 - val_loss: 0.7927
Epoch 63/150
27/27 - 5s - 169ms/step - accuracy: 0.8652 - loss: 0.4072 -
val_accuracy: 0.7897 - val_loss: 0.7153
Epoch 64/150
27/27 - 5s - 184ms/step - accuracy: 0.8664 - loss: 0.4013 -
val_accuracy: 0.7804 - val_loss: 0.8035
Epoch 65/150
27/27 - 5s - 197ms/step - accuracy: 0.8699 - loss: 0.3996 -
val_accuracy: 0.7844 - val_loss: 0.7501
Epoch 66/150
27/27 - 5s - 203ms/step - accuracy: 0.8653 - loss: 0.4059 -
val_accuracy: 0.7630 - val_loss: 0.8889
Epoch 67/150
27/27 - 6s - 215ms/step - accuracy: 0.8654 - loss: 0.4017 -
val_accuracy: 0.7917 - val_loss: 0.7333
Epoch 68/150
27/27 - 6s - 211ms/step - accuracy: 0.8719 - loss: 0.3850 -
val_accuracy: 0.7810 - val_loss: 0.8211
Epoch 69/150
27/27 - 5s - 183ms/step - accuracy: 0.8746 - loss: 0.3756 -
val_accuracy: 0.7857 - val_loss: 0.7892
Epoch 70/150
27/27 - 6s - 204ms/step - accuracy: 0.8752 - loss: 0.3725 -
val_accuracy: 0.7804 - val_loss: 0.8320
Epoch 71/150
27/27 - 5s - 190ms/step - accuracy: 0.8752 - loss: 0.3778 -
```

```
val_accuracy: 0.7684 - val_loss: 0.8583
Epoch 72/150
27/27 - 4s - 160ms/step - accuracy: 0.8774 - loss: 0.3638 -
val_accuracy: 0.7857 - val_loss: 0.8245
Epoch 73/150
27/27 - 4s - 156ms/step - accuracy: 0.8786 - loss: 0.3763 -
val_accuracy: 0.7677 - val_loss: 0.8590
Epoch 74/150
27/27 - 4s - 157ms/step - accuracy: 0.8768 - loss: 0.3613 -
val_accuracy: 0.7917 - val_loss: 0.7976
Epoch 75/150
27/27 - 4s - 158ms/step - accuracy: 0.8799 - loss: 0.3624 -
val_accuracy: 0.7917 - val_loss: 0.7408
Epoch 76/150
27/27 - 4s - 155ms/step - accuracy: 0.8818 - loss: 0.3625 -
val_accuracy: 0.7837 - val_loss: 0.8286
Epoch 77/150
27/27 - 4s - 159ms/step - accuracy: 0.8868 - loss: 0.3443 -
val_accuracy: 0.7984 - val_loss: 0.7459
Epoch 78/150
27/27 - 4s - 158ms/step - accuracy: 0.8903 - loss: 0.3315 -
val_accuracy: 0.7617 - val_loss: 0.9773
Epoch 79/150
27/27 - 4s - 164ms/step - accuracy: 0.8858 - loss: 0.3473 -
val_accuracy: 0.7690 - val_loss: 0.8803
Epoch 80/150
27/27 - 5s - 173ms/step - accuracy: 0.8900 - loss: 0.3346 -
val_accuracy: 0.7657 - val_loss: 0.9110
Epoch 81/150
27/27 - 5s - 186ms/step - accuracy: 0.8925 - loss: 0.3238 -
val_accuracy: 0.7810 - val_loss: 0.8663
Epoch 82/150
27/27 - 5s - 182ms/step - accuracy: 0.8854 - loss: 0.3476 -
val_accuracy: 0.7991 - val_loss: 0.7793
Epoch 83/150
27/27 - 4s - 159ms/step - accuracy: 0.8901 - loss: 0.3255 -
val_accuracy: 0.7844 - val_loss: 0.8848
Epoch 84/150
27/27 - 4s - 164ms/step - accuracy: 0.8912 - loss: 0.3342 -
val_accuracy: 0.7824 - val_loss: 0.8661
Epoch 85/150
27/27 - 4s - 159ms/step - accuracy: 0.8888 - loss: 0.3384 -
val_accuracy: 0.8017 - val_loss: 0.7707
Epoch 86/150
27/27 - 4s - 163ms/step - accuracy: 0.8927 - loss: 0.3270 -
val_accuracy: 0.7817 - val_loss: 0.7707
Epoch 87/150
27/27 - 4s - 159ms/step - accuracy: 0.8889 - loss: 0.3365 -
val_accuracy: 0.7710 - val_loss: 0.8975
```

```
Epoch 88/150
27/27 - 4s - 150ms/step - accuracy: 0.8916 - loss: 0.3271 -
val_accuracy: 0.7590 - val_loss: 0.9255
Epoch 89/150
27/27 - 4s - 151ms/step - accuracy: 0.8952 - loss: 0.3190 -
val_accuracy: 0.7850 - val_loss: 0.8629
Epoch 90/150
27/27 - 4s - 147ms/step - accuracy: 0.8946 - loss: 0.3180 -
val_accuracy: 0.8064 - val_loss: 0.8065
Epoch 91/150
27/27 - 4s - 160ms/step - accuracy: 0.8969 - loss: 0.3163 -
val_accuracy: 0.7710 - val_loss: 0.9495
Epoch 92/150
27/27 - 4s - 154ms/step - accuracy: 0.8983 - loss: 0.2999 -
val_accuracy: 0.7971 - val_loss: 0.8616
Epoch 93/150
27/27 - 4s - 154ms/step - accuracy: 0.8971 - loss: 0.3004 -
val_accuracy: 0.7764 - val_loss: 0.8881
Epoch 94/150
27/27 - 4s - 150ms/step - accuracy: 0.9020 - loss: 0.2957 -
val_accuracy: 0.7951 - val_loss: 0.8460
Epoch 95/150
27/27 - 4s - 154ms/step - accuracy: 0.9043 - loss: 0.2956 -
val_accuracy: 0.7603 - val_loss: 1.0459
Epoch 96/150
27/27 - 4s - 154ms/step - accuracy: 0.8986 - loss: 0.3001 -
val_accuracy: 0.7677 - val_loss: 0.9690
Epoch 97/150
27/27 - 4s - 153ms/step - accuracy: 0.9023 - loss: 0.2899 -
val_accuracy: 0.8011 - val_loss: 0.8650
Epoch 98/150
27/27 - 4s - 160ms/step - accuracy: 0.9052 - loss: 0.2915 -
val_accuracy: 0.7931 - val_loss: 0.8822
Epoch 99/150
27/27 - 4s - 166ms/step - accuracy: 0.9050 - loss: 0.2921 -
val_accuracy: 0.8011 - val_loss: 0.8016
Epoch 100/150
27/27 - 4s - 165ms/step - accuracy: 0.9052 - loss: 0.2918 -
val_accuracy: 0.7744 - val_loss: 1.0008
Epoch 101/150
27/27 - 4s - 165ms/step - accuracy: 0.9003 - loss: 0.2972 -
val_accuracy: 0.7824 - val_loss: 0.8160
Epoch 102/150
27/27 - 4s - 160ms/step - accuracy: 0.9035 - loss: 0.2906 -
val_accuracy: 0.8044 - val_loss: 0.8067
Epoch 103/150
27/27 - 5s - 176ms/step - accuracy: 0.9075 - loss: 0.2753 -
val_accuracy: 0.7543 - val_loss: 1.0394
Epoch 104/150
```

```
27/27 - 4s - 165ms/step - accuracy: 0.9147 - loss: 0.2701 -
val_accuracy: 0.7897 - val_loss: 1.0063
Epoch 105/150
27/27 - 4s - 165ms/step - accuracy: 0.9064 - loss: 0.2882 -
val_accuracy: 0.7951 - val_loss: 0.8891
Epoch 106/150
27/27 - 4s - 166ms/step - accuracy: 0.9109 - loss: 0.2777 -
val_accuracy: 0.8084 - val_loss: 0.8777
Epoch 107/150
27/27 - 5s - 171ms/step - accuracy: 0.9067 - loss: 0.2890 -
val_accuracy: 0.7830 - val_loss: 0.9450
Epoch 108/150
27/27 - 4s - 162ms/step - accuracy: 0.9083 - loss: 0.2733 -
val_accuracy: 0.7991 - val_loss: 0.9087
Epoch 109/150
27/27 - 5s - 172ms/step - accuracy: 0.9121 - loss: 0.2700 -
val_accuracy: 0.7837 - val_loss: 0.9496
Epoch 110/150
27/27 - 5s - 175ms/step - accuracy: 0.9146 - loss: 0.2604 -
val_accuracy: 0.8044 - val_loss: 0.8696
Epoch 111/150
27/27 - 5s - 174ms/step - accuracy: 0.9041 - loss: 0.2839 -
val_accuracy: 0.8031 - val_loss: 0.8606
Epoch 112/150
27/27 - 5s - 181ms/step - accuracy: 0.9090 - loss: 0.2801 -
val_accuracy: 0.7911 - val_loss: 0.8998
Epoch 113/150
27/27 - 5s - 173ms/step - accuracy: 0.9086 - loss: 0.2721 -
val_accuracy: 0.7637 - val_loss: 0.9702
Epoch 114/150
27/27 - 4s - 165ms/step - accuracy: 0.9136 - loss: 0.2717 -
val_accuracy: 0.7971 - val_loss: 0.9202
Epoch 115/150
27/27 - 4s - 149ms/step - accuracy: 0.9127 - loss: 0.2700 -
val_accuracy: 0.7844 - val_loss: 0.9619
Epoch 116/150
27/27 - 5s - 169ms/step - accuracy: 0.9152 - loss: 0.2583 -
val_accuracy: 0.7657 - val_loss: 1.1282
Epoch 117/150
27/27 - 4s - 156ms/step - accuracy: 0.9122 - loss: 0.2629 -
val_accuracy: 0.8017 - val_loss: 0.9393
Epoch 118/150
27/27 - 4s - 166ms/step - accuracy: 0.9170 - loss: 0.2544 -
val_accuracy: 0.7904 - val_loss: 0.8739
Epoch 119/150
27/27 - 4s - 165ms/step - accuracy: 0.9182 - loss: 0.2626 -
val_accuracy: 0.7904 - val_loss: 0.9871
Epoch 120/150
27/27 - 5s - 167ms/step - accuracy: 0.9192 - loss: 0.2539 -
```

```
val_accuracy: 0.7617 - val_loss: 1.1087
Epoch 121/150
27/27 - 5s - 174ms/step - accuracy: 0.9164 - loss: 0.2548 -
val_accuracy: 0.7931 - val_loss: 0.9568
Epoch 122/150
27/27 - 4s - 155ms/step - accuracy: 0.9156 - loss: 0.2571 -
val_accuracy: 0.7904 - val_loss: 0.9497
Epoch 123/150
27/27 - 4s - 164ms/step - accuracy: 0.9148 - loss: 0.2593 -
val_accuracy: 0.7877 - val_loss: 0.9758
Epoch 124/150
27/27 - 4s - 151ms/step - accuracy: 0.9138 - loss: 0.2649 -
val_accuracy: 0.7670 - val_loss: 1.0648
Epoch 125/150
27/27 - 4s - 150ms/step - accuracy: 0.9194 - loss: 0.2497 -
val_accuracy: 0.8037 - val_loss: 0.8803
Epoch 126/150
27/27 - 4s - 147ms/step - accuracy: 0.9220 - loss: 0.2413 -
val_accuracy: 0.8131 - val_loss: 0.8682
Epoch 127/150
27/27 - 5s - 170ms/step - accuracy: 0.9236 - loss: 0.2344 -
val_accuracy: 0.7937 - val_loss: 0.9537
Epoch 128/150
27/27 - 5s - 174ms/step - accuracy: 0.9193 - loss: 0.2450 -
val_accuracy: 0.7764 - val_loss: 1.0526
Epoch 129/150
27/27 - 4s - 158ms/step - accuracy: 0.9128 - loss: 0.2739 -
val_accuracy: 0.7677 - val_loss: 1.2130
Epoch 130/150
27/27 - 4s - 153ms/step - accuracy: 0.9150 - loss: 0.2550 -
val_accuracy: 0.7931 - val_loss: 0.9776
Epoch 131/150
27/27 - 4s - 161ms/step - accuracy: 0.9192 - loss: 0.2419 -
val_accuracy: 0.7904 - val_loss: 0.9403
Epoch 132/150
27/27 - 4s - 150ms/step - accuracy: 0.9215 - loss: 0.2380 -
val_accuracy: 0.7904 - val_loss: 1.0253
Epoch 133/150
27/27 - 4s - 151ms/step - accuracy: 0.9228 - loss: 0.2350 -
val_accuracy: 0.7884 - val_loss: 1.0296
Epoch 134/150
27/27 - 4s - 158ms/step - accuracy: 0.9216 - loss: 0.2444 -
val_accuracy: 0.7904 - val_loss: 1.0092
Epoch 135/150
27/27 - 4s - 151ms/step - accuracy: 0.9237 - loss: 0.2352 -
val_accuracy: 0.7837 - val_loss: 1.0535
Epoch 136/150
27/27 - 4s - 149ms/step - accuracy: 0.9174 - loss: 0.2599 -
val_accuracy: 0.7904 - val_loss: 0.9712
```

```
Epoch 137/150
27/27 - 4s - 161ms/step - accuracy: 0.9183 - loss: 0.2428 -
val_accuracy: 0.7857 - val_loss: 1.1310
Epoch 138/150
27/27 - 4s - 160ms/step - accuracy: 0.9281 - loss: 0.2276 -
val_accuracy: 0.8057 - val_loss: 0.9319
Epoch 139/150
27/27 - 5s - 169ms/step - accuracy: 0.9212 - loss: 0.2345 -
val_accuracy: 0.7850 - val_loss: 1.0416
Epoch 140/150
27/27 - 5s - 167ms/step - accuracy: 0.9213 - loss: 0.2520 -
val_accuracy: 0.7971 - val_loss: 0.9573
Epoch 141/150
27/27 - 5s - 179ms/step - accuracy: 0.9280 - loss: 0.2180 -
val_accuracy: 0.8151 - val_loss: 0.9279
Epoch 142/150
27/27 - 5s - 188ms/step - accuracy: 0.9308 - loss: 0.2097 -
val_accuracy: 0.8064 - val_loss: 0.9680
Epoch 143/150
27/27 - 4s - 151ms/step - accuracy: 0.9261 - loss: 0.2255 -
val_accuracy: 0.7937 - val_loss: 1.0357
Epoch 144/150
27/27 - 4s - 147ms/step - accuracy: 0.9167 - loss: 0.2528 -
val_accuracy: 0.7877 - val_loss: 1.1246
Epoch 145/150
27/27 - 4s - 161ms/step - accuracy: 0.9257 - loss: 0.2353 -
val_accuracy: 0.7964 - val_loss: 0.9948
Epoch 146/150
27/27 - 4s - 154ms/step - accuracy: 0.9211 - loss: 0.2358 -
val_accuracy: 0.7697 - val_loss: 1.1845
Epoch 147/150
27/27 - 4s - 150ms/step - accuracy: 0.9241 - loss: 0.2348 -
val_accuracy: 0.7837 - val_loss: 1.0375
Epoch 148/150
27/27 - 4s - 158ms/step - accuracy: 0.9228 - loss: 0.2358 -
val_accuracy: 0.8057 - val_loss: 0.9305
Epoch 149/150
27/27 - 5s - 169ms/step - accuracy: 0.9155 - loss: 0.2606 -
val_accuracy: 0.7817 - val_loss: 1.0634
Epoch 150/150
27/27 - 5s - 170ms/step - accuracy: 0.9232 - loss: 0.2357 -
val_accuracy: 0.7944 - val_loss: 0.9806
```

```python
train_loss, train_acc = model.evaluate(x_train, y_train, verbose = 2)
print('\nTrain Accuracy :', train_acc)
print('\nTrain Loss :', train_loss)
```

```
422/422 - 2s - 5ms/step - accuracy: 0.9625 - loss: 0.1180

Train Accuracy : 0.9625287652015686
```

```
Train Loss : 0.11796867102384567
```

```
test_loss, test_acc = model.evaluate(x_test, y_test, verbose = 2)
print('\nTest Accuracy :', test_acc)
print('\nTest Loss :', test_loss)
```

```
47/47 - 0s - 10ms/step - accuracy: 0.7944 - loss: 0.9806

Test Accuracy : 0.7943925261497498

Test Loss : 0.9805899262428284
```

```
print(f"CNN Model\n")
print(f"Training Accuracy: {round(train_acc * 100, 4)}% \nTrain Loss:
{round(train_loss, 4)}\n")
print(f"Testing Accuracy: {round(test_acc * 100, 4)}% \nTest Loss:
{round(test_loss, 4)}")
```

```
CNN Model

Training Accuracy: 96.2529%
Train Loss: 0.118

Testing Accuracy: 79.4393%
Test Loss: 0.9806
```

```
# pick a sample to predict from the test set
x_to_predict = x_test[180]
y_to_predict = y_test[180]

# predict sample
print(predict(model, x_to_predict, y_to_predict))
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 166ms/step

Actual Label: C:\Users\Shaivya\Desktop\Data\genres_original\jazz
Predicted Label: ['C:\\Users\\Shaivya\\Desktop\\Data\\
genres_original\\jazz']
None
```