ICT2202 - Digital Forensic

Audio Digger User Manual

SH4US

2002949	Chua Ying Wei, Dexter
2001759	Sim Kai Ching
2002293	Rachel Wong
2001222	Zulekah Khan Binte Salahuddin Khan

Table of Contents

List of Abbreviations **3**

1. About **4**

2. Prerequisites **4**
2.1 Python 3.8 4
2.2 FFMPEG 4
2.3 Requirement.txt 5

3. Functions **6**
3.1 Find & Export Information of Audio Files 8
3.1.1 ID3 Tags in MP3 Files 8
3.1.2 Header Information of WAV Files 8
3.1.3 Exporting from a Directory of Audio Files 9
3.2 Create and Export Spectrograms 10
3.2.1 Exporting out a Single Spectrogram 10
3.2.2 Exporting out All Spectrograms (Separate) 11
3.2.3 Exporting out All Spectrograms (Summary) 12
3.2.4 Changing Colour of the Spectrogram 12
3.2.5 Trimming the Spectrogram according to X Axis 13
3.2.6 Trimming the Spectrogram according to Y Axis 14
3.3 Display and Export Hex Dump of Audio Files 15
3.3.1 Exporting out Hex dump for a Audio File 15
3.3.2 Exporting out Hex dump from a Directory of Audio Files 16
3.4 Display and Export Bin Dump of Audio Files 17
3.4.1 Exporting out Bin dump for a Audio File 17
3.4.2 Exporting out Bin dump from a Directory of Audio Files 18

4. Error Handling **18**
4.1 Generic Path Error Handling 18
4.2 Header Information Error Handling 19
4.3 Spectrogram Error Handling 20
4.4 Hex and Bin Error Handling 21

List of Abbreviations

The Following Table describes the Abbreviations used in this document

Abbreviations	Full Meaning
FFmpeg	Fast Forward MPEG (Motion Picture Experts Group) - An open-source software project consisting of a suite of libraries and programs for handling video, audio, and other multimedia files.
CLI	Command Line Interface - Processes commands to a computer program in the form of lines of text.
.mp3	MP3 (formally MPEG-1 Audio Layer III /MPEG-2 Audio Layer III) - A coding format for digital audio developed largely by the Fraunhofer Society in Germany.
.wav	Waveform Audio File Format - An audio file format standard, developed by IBM and Microsoft, for storing an audio bitstream on PCs.

1. About

AudioDigger is a Windows 10 based program and focuses on two types of audio files, namely, MP3 and WAV files.You can input any MP3 or WAV file and run the program to investigate further into the files with functions such as checking audio information embedded in the header and tags along to exporting spectrograms images.

To get AudioDigger, please refer to our Github Repository (https://github.com/rachwrong/ICT2202-SH4US) and download/pull from our main branch.

2. Prerequisites

For AudioDigger to run optimally with minimum errors, you are encouraged to refer to the following section on what are the prerequisites of our program and install any packages that may be lacking in your system.

2.1 Python 3.8

This program requires python 3.8 (preferably 3.8.10) to be installed in your Windows machine. To install python 3.8, please refer to the official Python website (https://www.python.org/downloads/windows/) and scroll down to the section for Python 3.8.10 to download the Windows installer.

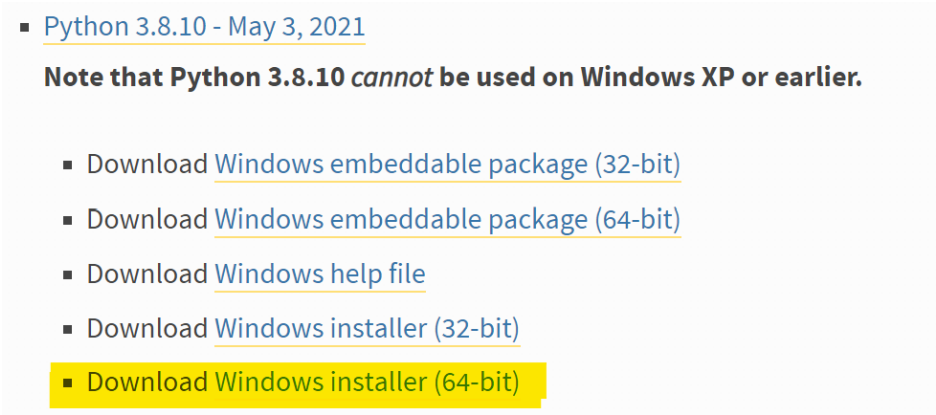

Figure 1 - Python 3.8 installer

2.2 FFMPEG

The program requires FFmpeg to be installed on your Windows machine. To install FFmpeg, please refer to the official FFmpeg website (https://www.ffmpeg.org/download.html) and click on the link under “Get packages & executable files - Windows EXE files”. You will only need to download the essentials.

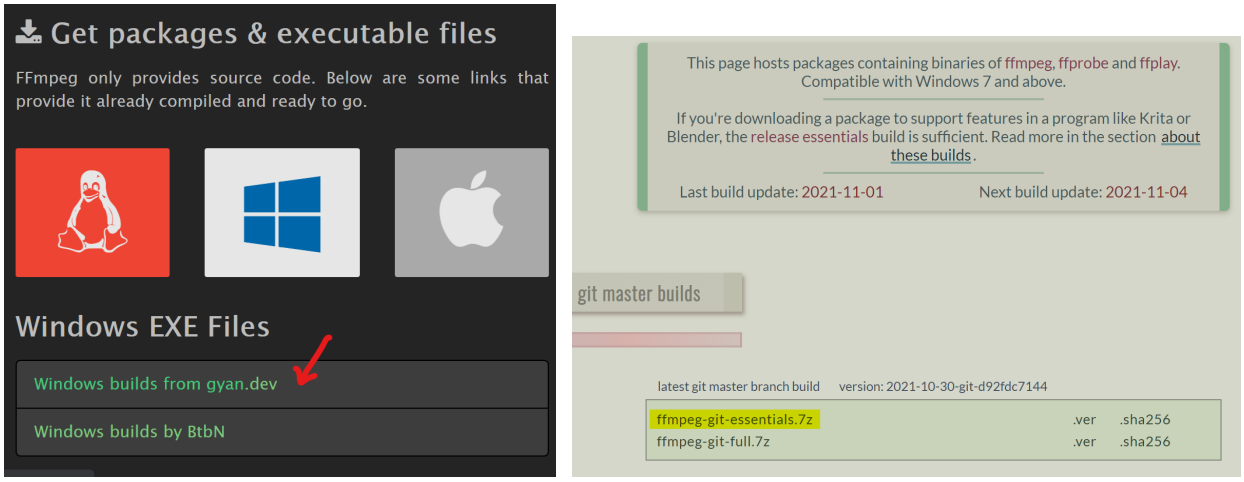

Figure 2 - 3 FFmpeg official downloadable packages

After downloading and extracting the 7zip file, ensure that FFmpeg is installed into the correct directories and have a valid system environment variable path in your machine.

2.3 Requirement.txt

We have provided a text file called “requirement.txt" containing all the libraries needed to run AudioDigger successfully in the Github repository. You can choose to either download this text file and run *pip install -r requirements.txt* in your CLI or installing the libraries manually with the following commands:

- ➔ pip install pydub
- ➔ pip install matplotlib
- ➔ pip install scipy
- ➔ pip install mutagen

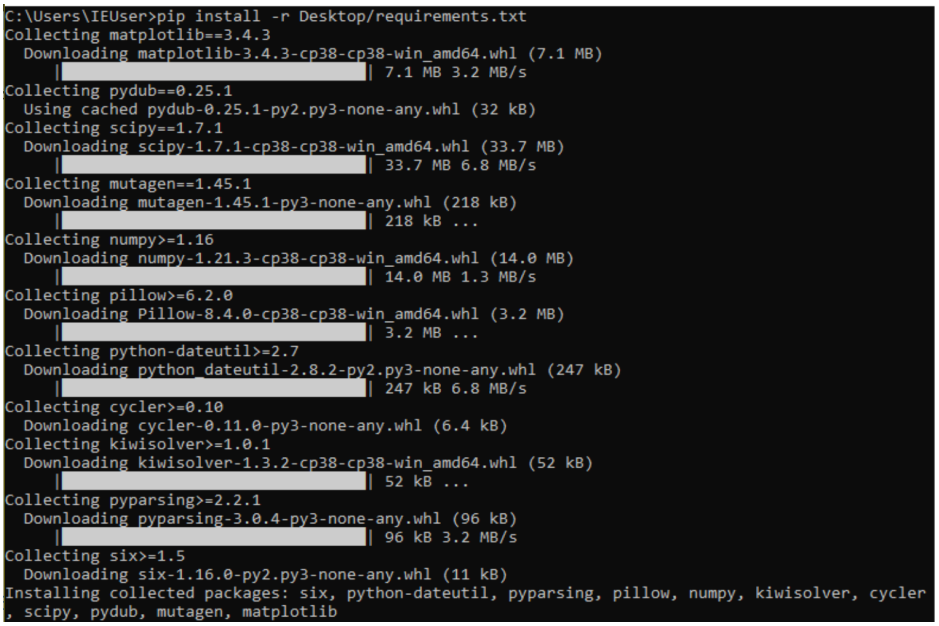

Figure - 4 Using requirements.txt to install the libraries

3. Functions

AudioDigger has 4 main features available. The following sections will explain and show examples on how to use them.

- ➔ 3.1 Display and Exporting of Header Information of MP3 and WAV audio files
- ➔ 3.2 Creating and Exporting of MP3 and WAV audio files Spectrograms
- ➔ 3.3 Exporting of Hexadecimal Dumps of MP3 and WAV audio files
- ➔ 3.4 Exporting of Binary Dumps of MP3 and WAV audio files

When you enter *python AudioDigger.py -h* in the CLI, you will be able to see the complete help documentation and the arguments taken by this program.

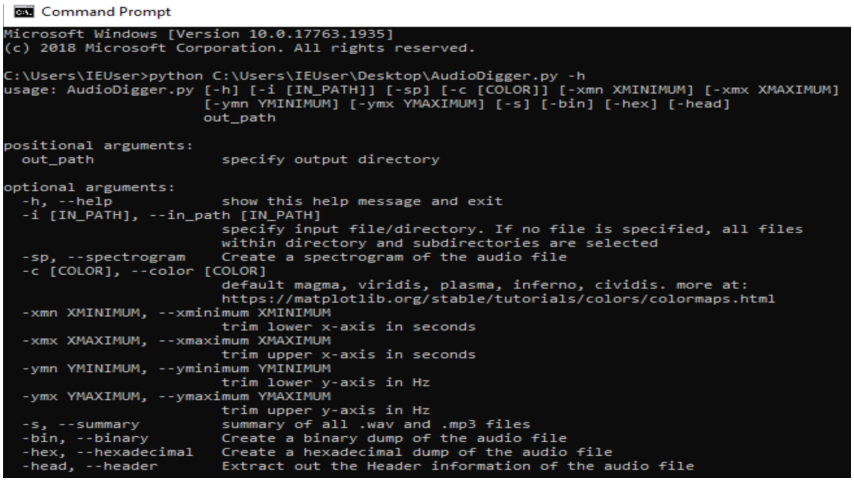

Figure - 5 Entering -h command in CLI

A table showing all the arguments is also provided below.

Argument	Full Argument	Description
-h	--help	Show this help message and exit
-i	--in_path	Specify input file/directory. If no file is specified, all files within directory and subdirectories are selected
-sp	--spectrogram	Create a spectrogram of the audio file
-c	--color	Default magma, viridis, plasma, inferno, cividis. more at: https://matplotlib.org/stable/tutorials/colors/colormaps.html
-xmn	--xminimum	Trim lower x-axis in seconds
-xmx	--xmaximum	Trim upper x-axis in seconds
-ymn	--yminimum	Trim lower y-axis in seconds
-ymx	--ymaximum	Trim upper y-axis in seconds
-s	--summary	Summary of all .wav and .mp3 files
-bin	--binary	Create a binary dump of the audio file
-hex	--hexadecimal	Create a hexadecimal dump of the audio file
-head	--header	Extract out the Header information of the audio file

3.1 Find & Export Information of Audio Files

This function allows you to view the information embedded in the audio file itself, usually in the header of the audio (in the case of .WAV file) or in the ID3 tags (in the case of .MP3 file).

The general argument to call this function is *python AudioDigger.py -i [in_path] [out_path] -head*

3.1.1 ID3 Tags in MP3 Files

The following example below will show how this function will work for MP3 audio files. In Figure 6, a user is running the *-header* command to check and export the information of a specific MP3 file. In Figure 7, the user can see the output after running this command. If the checking and exportation of that audio file is successful, a success message will be shown at the end.

```
C:\Users\IEUser>python C:\Users\IEUser\Desktop\AudioDigger.py -i C:\Users\IEUser\Desktop\Import\ImpactModerat
o.mp3 C:\Users\IEUser\Desktop\Export -head
MPEG 1 layer 3, 224000 bps (CBR, LAME 3.99.1+, -b 224), 32000 Hz, 2 chn, 27.19 seconds (audio/mp3)
TALB=YouTube Audio Library
TCON=Cinematic
TIT2=Impact Moderato
TPE1=Kevin MacLeod
ImpactModerato Header Information is exported!
```

Figure - 6 using -head command in CLI

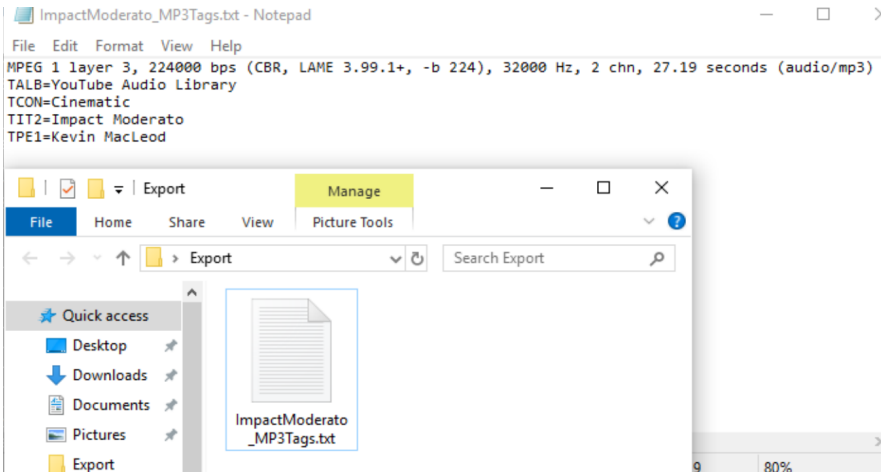

Figure - 7 Exporting of information

3.1.2 Header Information of WAV Files

The following example below will show how this function will work for WAV audio files. In Figure 8, a user is running the *-header* command to check and export the information of a specific WAV file. In Figure 9, the user can see the output after running this command. If the checking and exportation of that audio file is successful, a success message will be shown at the end.

```
C:\Users\IEUser>python C:\Users\IEUser\Desktop\AudioDigger.py -i C:\Users\IEUser\Desktop\Import
\PinkPanther.wav C:\Users\IEUser\Desktop\Export -head
This is the header information for PinkPanther :
ChunkID= b'RIFF'
FileSize= 1323044
Format= b'WAVE'
SubChunk1ID= b'fmt '
SubChunk1Size= 16
AudioFormat= 1
NumChannels= 1
SampleRate= 22050
ByteRate= 44100
BlockAlign= 2
BitsPerSample= 16
SubChunk2ID= b'data'
SubChunk2Size= 1323000
PinkPanther Header Information is exported!
```

Figure - 8 using -head command in CLI

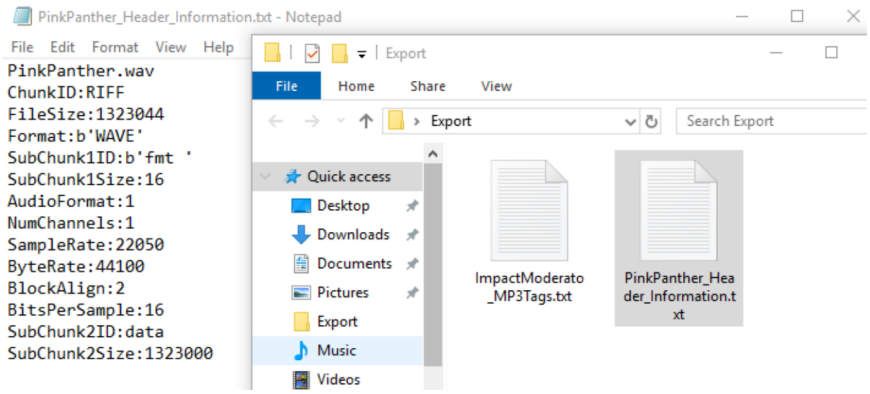

Figure - 9 Exporting of information

3.1.3 Exporting from a Directory of Audio Files

The following example below will show how this function will work for exporting the information of multiple audio files in a directory.

The argument to call this function is *python AudioDigger.py -i [in_path] [out_path] -head -s,* in which the [in_path] should end with the directory containing all the audio files.

In Figure 10, a user is running the *-head -s* command to check and export the information of all valid MP3 and WAV files. In Figure 11, the user can see the output after running this command. If the checking and exportation of all valid audio files is successful, a success message will be shown at the end.

```
C:\Users\IEUser>python Desktop/AudioDigger.py -i C:\Users\IEUser\Desktop\Import C:\Users\IEUser
\Desktop\Export -head -s
some-random-song is not a valid mp3 file, High probability of being tempered with.
Testfile.txt is not a valid wav audio file. Header information will not be added in Summary fil
e!
Header information Summary file is created successfully!
```

Figure - 10 using -head -s command in CLI

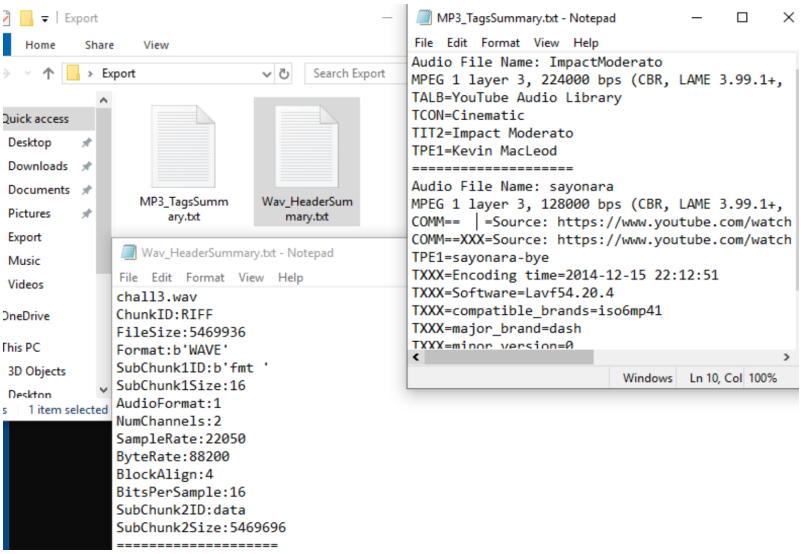

Figure - 11 Exporting of information for all valid audio files

3.2 Create and Export Spectrograms

This function allows you to export and view the spectrograms of valid audio files.

The general argument to call this function is *python AudioDigger.py -i [in_path] [out_path] -sp*

3.2.1 Exporting out a Single Spectrogram

The following example below will show how this function will work for a single audio file. In Figure 12, a user is running the *-sp* command to export the spectrogram into a .PNG file. In Figure 13, the user can see the output after running this command. If the exportation of that audio file is successful, the user will see the spectragram in the specified out path directory.

```
C:\Users\IEUser>python Desktop/AudioDigger.py -i C:\Users\IEUser\Desktop\Import\some-random-son
g.mp3 C:\Users\IEUser\Desktop\Export -sp
```

Figure - 12 using -sp command in CLI

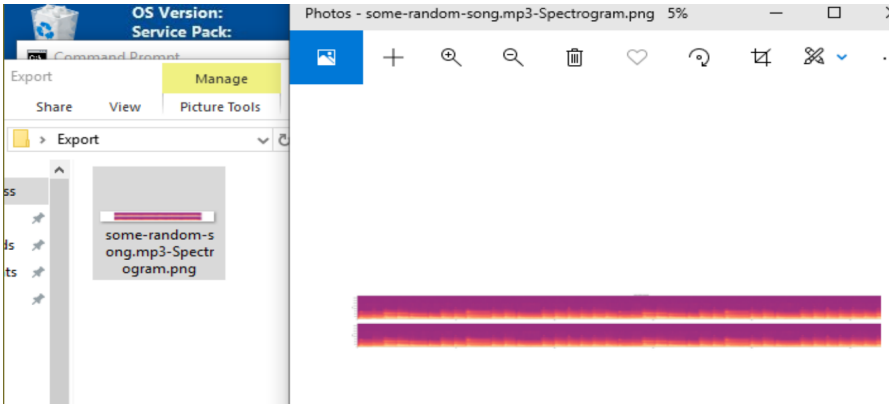

Figure - 13 Exporting of spectrogram for the audio file

3.2.2 Exporting out All Spectrograms (Separate)

The following example below will show how this function will work for exporting multiple spectrograms of audio files in a directory.

The argument to call this function is *python AudioDigger.py -i [in_path] [out_path] -sp,* in which the [in_path] should end with the directory containing all the audio files.

In Figure 14, a user is running the *-sp* command to export the spectrogram into a .PNG file. In Figure 15, the user can see the output after running this command. If the exportation of the audio files is successful, the user will see the exported spectrograms in the specified out path directory.

```
C:\Users\IEUser>python Desktop/AudioDigger.py -i C:\Users\IEUser\Desktop\Import C:\Users\IEUser\D
esktop\Export -sp
```

Figure - 14 using -sp command in CLI

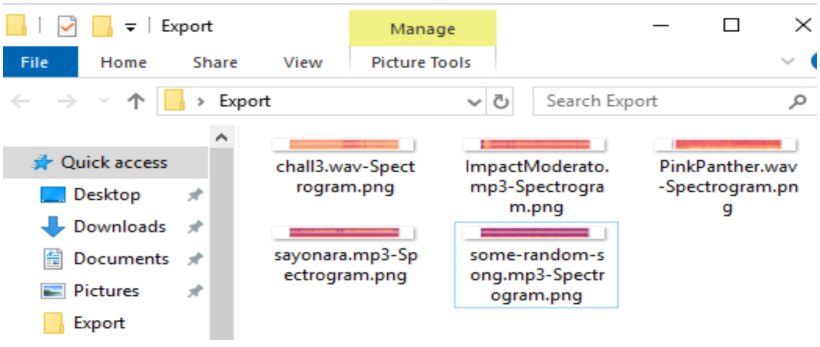

Figure - 15 Exported spectrograms in out directory

3.2.3 Exporting out All Spectrograms (Summary)

The following example below will show how this function will work for exporting a spectrogram summary of multiple audio files in a directory.

The argument to call this function is *python AudioDigger.py -i [in_path] [out_path] -sp -s,* in which the [in_path] should end with the directory containing all the audio files.

In Figure 16, a user is running the *-sp -s* command to export the summary spectrogram into a .PNG file. In Figure 17, the user can see the output after running this command. If the exportation of the audio files is successful, the user will see the summary spectragram in the specified out path directory.

```
C:\Users\IEUser>python Desktop/AudioDigger.py -i C:\Users\IEUser\Desktop\Import C:\Users\IEUser
\Desktop\Export -sp -s
```

Figure - 16 using -sp command in CLI

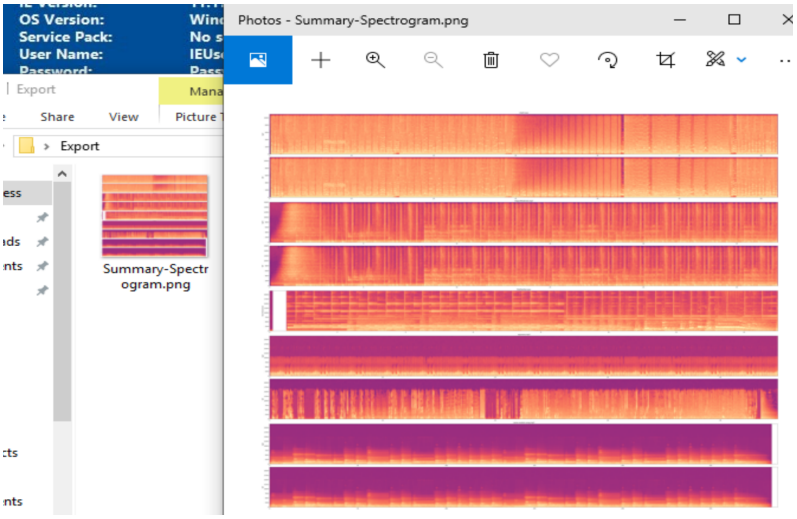

Figure - 17 Exported Summary spectrogram in out directory

3.2.4 Changing Colour of the Spectrogram

The following example below will show how this function will work for changing the color of the spectrogram.

The argument to call this function is *python AudioDigger.py -i [in_path] -c <color> [out_path] -sp,* in which the <color> is user defined.

In Figure 18, a user is running the *-c* command to change the color of the spectrogram. The default colour used for the spectrogram is “magma” and you can find out more about the colors and their name in the Matplotlib Colormaps (https://matplotlib.org/stable/tutorials/colors/colormaps.html). In Figure 19, the user can see the

output after running this command. If the exportation of that audio file is successful, the user will see the color of the spectragram changed to their specific color.

```
C:\Users\IEUser>python Desktop/AudioDigger.py -i C:\Users\IEUser\Desktop\Import\some-random-son
g.mp3 -c viridis C:\Users\IEUser\Desktop\Export -sp
```

Figure - 18 using -c command in CLI

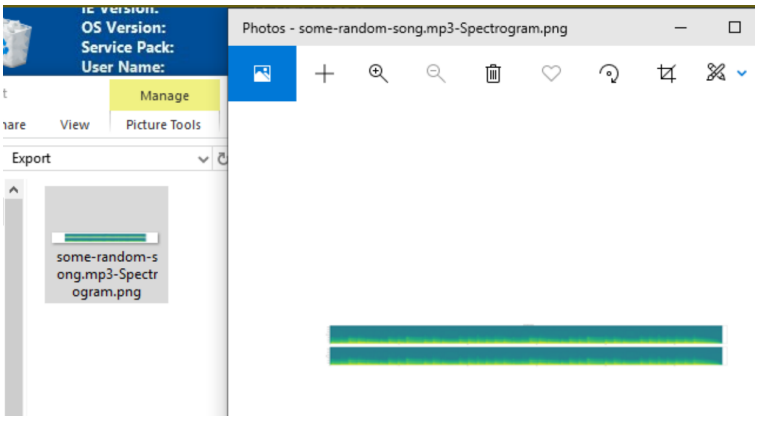

Figure - 19 Exported spectrogram in a different colour

3.2.5 Trimming the Spectrogram according to X Axis

The following example below will show how this function will work for trimming the X axis of the output in the exported spectrogram.

The argument to call this function is *python AudioDigger.py -i [in_path] -xmn <number> -xmx <number> [out_path] -sp,* in which the [in_path] should end with the directory containing all the audio files.

In Figure 20, a user is running the *-xmn -xmx* command to trim the spectrogram. In Figure 21, the user can see the output after running this command. If the exportation and trimming of the audio files is successful, the user will see the edited spectragram in the directory.

```
C:\Users\IEUser>python Desktop/AudioDigger.py -i C:\Users\IEUser\Desktop\Import\some-random-son
g.mp3 -xmn 5 -xmx 40 C:\Users\IEUser\Desktop\Export -sp
```

Figure - 20 using -xmn -xmx command in CLI

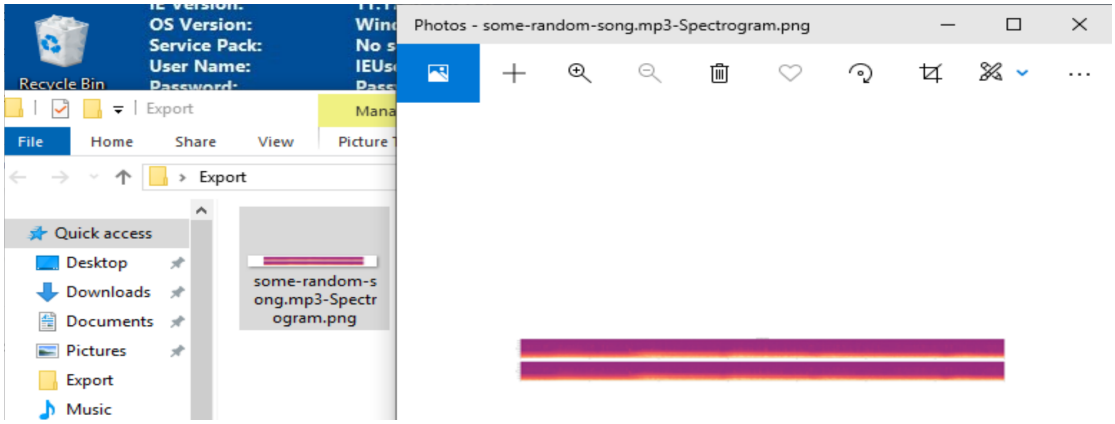

Figure - 21 Exported Spectrogram after it is trimmed

In addition, you can specify whether you would like to only trim the upper or lower X axis in the spectrogram.

The argument to trim the upper X axis is *python AudioDigger.py -i [in_path] -xmx <number> [out_path] -sp* (Figure 22), while the argument to trim the lower X axis is *python AudioDigger.py -i [in_path] -xmn <number> [out_path] -sp* (Figure 23)

```
C:\Users\IEUser>python Desktop/AudioDigger.py -i C:\Users\IEUser\Desktop\Import\some-random-song.
mp3 -xmx 40 C:\Users\IEUser\Desktop\Export -sp
```

Figure - 22 using -xmx command in CLI

```
C:\Users\IEUser>python Desktop/AudioDigger.py -i C:\Users\IEUser\Desktop\Import\some-random-song.
mp3 -xmn 40 C:\Users\IEUser\Desktop\Export -sp
```

Figure - 23 using -xmn command in CLI

3.2.6 Trimming the Spectrogram according to Y Axis

The following example below will show how this function will work for trimming the Y axis of the output in the exported spectrogram.

The argument to call this function is *python AudioDigger.py -i [in_path] -ymn <number> -ymx <number> [out_path] -sp,* in which the [in_path] should end with the directory containing all the audio files.

In Figure 24, a user is running the *-ymn -ymx* command to trim the spectrogram. In Figure 25, the user can see the output after running this command. If the exportation and trimming of the audio files is successful, the user will see the edited spectragram in the directory.

```
C:\Users\IEUser>python Desktop/AudioDigger.py -i C:\Users\IEUser\Desktop\Import\sayonara.mp3 -ymx
 7000 -ymn 2000 C:\Users\IEUser\Desktop\Export -sp
```

Figure - 24 using -ymn -ymx command in CLI

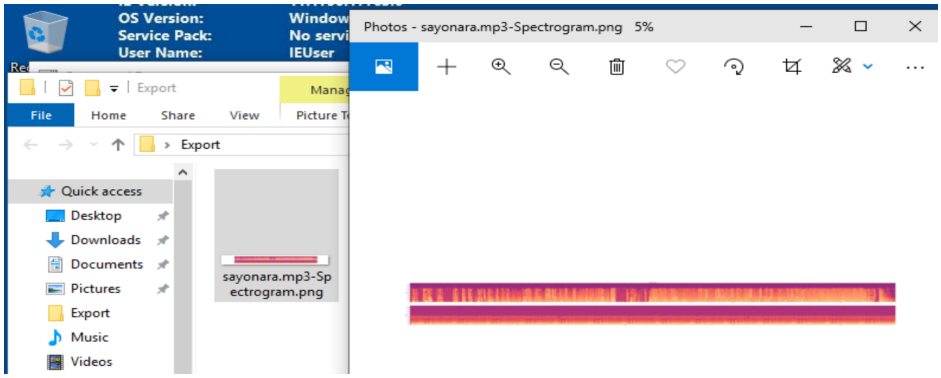

Figure - 25 Exported Spectrogram after it is trimmed

In addition, you can specify whether you would like to only trim the upper or lower Y axis in the spectrogram.

The argument to trim the upper Y axis is *python AudioDigger.py -i [in_path] -ymx <number> [out_path] -sp* (Figure 26), while the argument to trim the lower Y axis is *python AudioDigger.py -i [in_path] -ymn <number> [out_path] -sp* (Figure 27).

```
C:\Users\IEUser>python Desktop/AudioDigger.py -i C:\Users\IEUser\Desktop\Import\sayonara.mp3 -y
mx 7000 C:\Users\IEUser\Desktop\Export -sp -s
```

Figure - 26 using -ymx command in CLI

```
C:\Users\IEUser>python Desktop/AudioDigger.py -i C:\Users\IEUser\Desktop\Import\sayonara.mp3 -ymn
 2000 C:\Users\IEUser\Desktop\Export -sp
```

Figure - 27 using -ymn command in CLI

3.3 Display and Export Hex Dump of Audio Files

This function allows you to export and view the hexadecimal dump of audio files.

The general argument to call this function is *python AudioDigger.py -i [in_path] [out_path] -hex*

3.3.1 Exporting out Hex dump for a Audio File

The following example below will show how this function will work for a single audio file. In Figure 28, a user is running the *-hex* command to export the hex dump into a .TXT file. In Figure 29, the user can see the output after running this command. If the exportation of that audio file is successful, the user will see the text file in the specified out path directory.

```
C:\Users\IEUser>python Desktop/AudioDigger.py -i C:\Users\IEUser\Desktop\Import\sayonara.mp3 C:
\Users\IEUser\Desktop\Export -hex
sayonara.mp3 hex dump is created!
```

Figure - 28 using -hex command in CLI

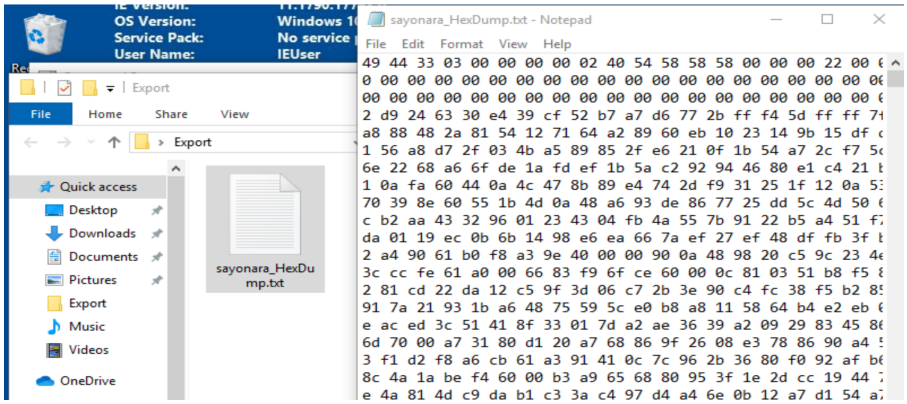

Figure - 29 Exporting of hex dumps

3.3.2 Exporting out Hex dump from a Directory of Audio Files

The following example below will show how this function will work for exporting the hex dumps of multiple audio files in a directory.

The argument to call this function is *python AudioDigger.py -i [in_path] [out_path] -hex -s,* in which the [in_path] should end with the directory containing all the audio files.

In Figure 30, a user is running the *-hex -s* command to check and export the hex dumps of all audio files in the directory. In Figure 31, the user can see the output after running this command. If the checking and exportation of all valid audio files is successful, a success message will be shown at the end.

```
C:\Users\IEUser>python Desktop/AudioDigger.py -i C:\Users\IEUser\Desktop\Import C:\Users\IEUser
\Desktop\Export -hex -s
chall3.wav hex dump is created!
ImpactModerato.mp3 hex dump is created!
PinkPanther.wav hex dump is created!
sayonara.mp3 hex dump is created!
some-random-song.mp3 hex dump is created!
Testfile.txt is not a mp3/wav audio file. Hex Dump will not be created!
```

Figure - 30 using -hex -s command in CLI

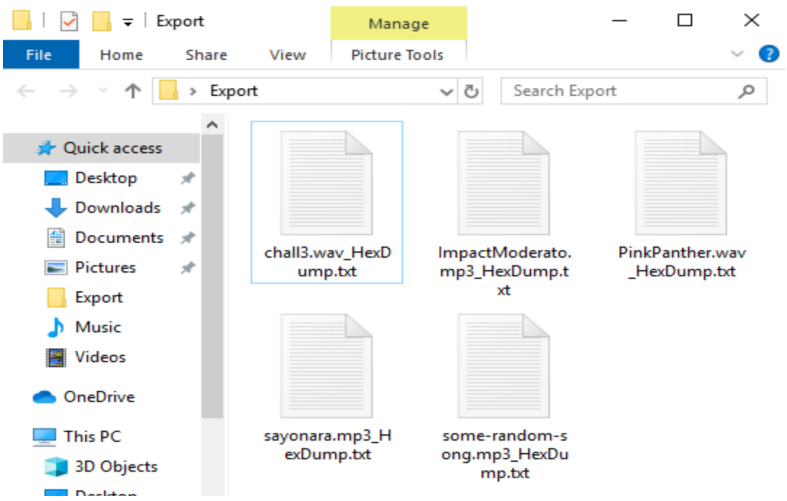

Figure - 31 Exporting of hex dumps

3.4 Display and Export Bin Dump of Audio Files

This function allows you to export and view the binary dump of audio files.

The general argument to call this function is *python AudioDigger.py -i [in_path] [out_path] -bin*

3.4.1 Exporting out Bin dump for a Audio File

The following example below will show how this function will work for a single audio file. In Figure 32, a user is running the *-bin* command to export the bin dump into a .TXT file. In Figure 33, the user can see the output after running this command. If the exportation of that audio file is successful, the user will see the text file in the specified out path directory.

```
C:\Users\IEUser>python Desktop/AudioDigger.py -i C:\Users\IEUser\Desktop\Import\sayonara.mp3 C:
\Users\IEUser\Desktop\Export -bin
sayonara.mp3 bin dump is created!
```

Figure - 32 using -bin command in CLI

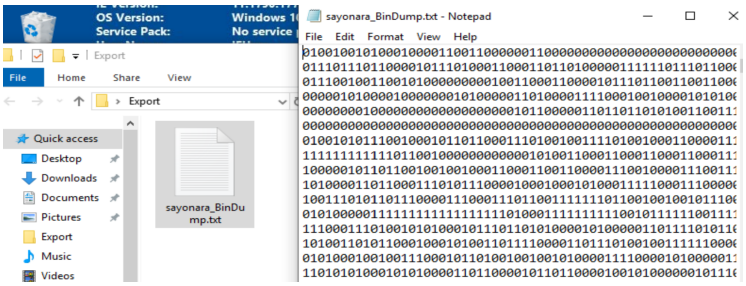

Figure - 33 Exporting of bin dump

3.4.2 Exporting out Bin dump from a Directory of Audio Files

The following example below will show how this function will work for exporting the bin dumps of multiple audio files in a directory.

The argument to call this function is *python AudioDigger.py -i [in_path] [out_path] -bin -s,* in which the [in_path] should end with the directory containing all the audio files.

In Figure 34, a user is running the *-bin -s* command to check and export the bin dumps of all audio files in the directory. In Figure 35, the user can see the output after running this command. If the checking and exportation of all valid audio files is successful, a success message will be shown at the end.

```
C:\Users\IEUser>python Desktop/AudioDigger.py -i C:\Users\IEUser\Desktop\Import C:\Users\IEUser
\Desktop\Export -bin -s
chall3.wav bin dump is created!
ImpactModerato.mp3 bin dump is created!
PinkPanther.wav bin dump is created!
sayonara.mp3 bin dump is created!
some-random-song.mp3 bin dump is created!
```

Figure - 34 using -bin -s command in CLI

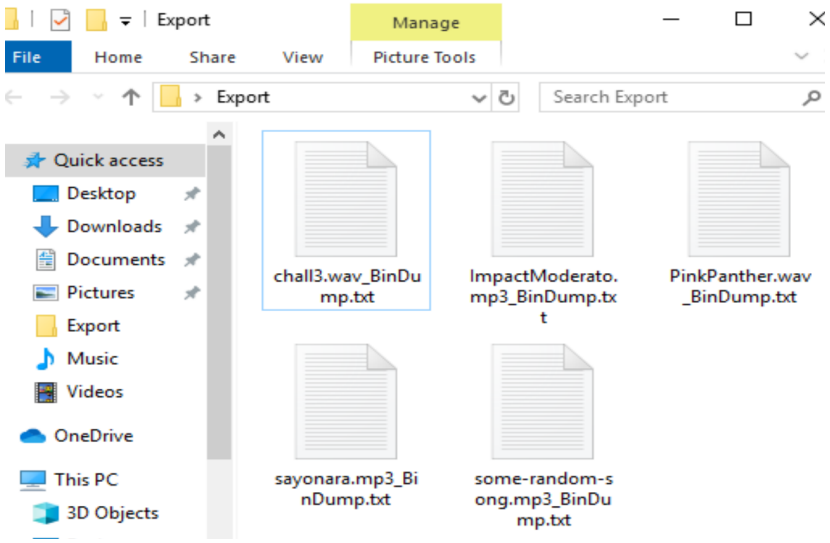

Figure - 35 Exporting of bin dumps

4. Error Handling

AudioDigger also has some inbuilt error handling features. The following section will explain and show examples on the types of error handling for each function.

4.1 Generic Path Error Handling

When you enter the wrong in path/out path, AudioDigger will display an error message warning about the invalid path as shown in Figure 36 and 37.

```
C:\Users\IEUser>python Desktop/AudioDigger.py -i C:\Users\IEUser\Desktop\Folder1 C:\Users\IEUser\
Desktop\Export -sp -s
Input file/directory does not exist.
```

Figure - 36 Error message for invalid in-path

```
C:\Users\IEUser>python Desktop/AudioDigger.py -i C:\Users\IEUser\Desktop\Import\PinkPanther.wav C
:\Users\IEUser\Desktop\Folder2 -sp
Output file/directory does not exist.
```

Figure - 37 Error message for invalid out-path

4.2 Header Information Error Handling

If you enter the argument *-s* with a defined file path as seen in Figure 38, AudioDigger will display an error message warning about the invalid path and the header information text file will not be created.

```
C:\Users\IEUser>python Desktop/AudioDigger.py -i C:\Users\IEUser\Desktop\Import\sayonara.mp3 C:
\Users\IEUser\Desktop\Export -head -s
Path is incorrect. Header Information Summary will not be created!
```

Figure - 38 Error message for using -s with defined filepath

If you try to export the header information for a directory without entering the argument *-s* as seen in Figure 39, AudioDigger will display an error message and the header information text file will not be created.

```
C:\Users\IEUser>python Desktop/AudioDigger.py -i C:\Users\IEUser\Desktop\Import C:\Users\IEUser
\Desktop\Export -head
Not a WAV file, please check your input again!
```

Figure - 39 Error message when -s is not used for directory path

If you try to use this function using a invalid file type as seen in Figure 40, AudioDigger will display an error message warning about the invalid file format and the header information text file will not be created.

```
C:\Users\IEUser>python Desktop/AudioDigger.py -i C:\Users\IEUser\Desktop\Import\Testfile.txt C:
\Users\IEUser\Desktop\Export -head
Not a WAV file, please check your input again!
```

Figure - 40 Error message for invalid file type

If you try to use this function using a invalid MP3 file type as seen in Figure 41, AudioDigger will display an error message warning about the invalid MP3 file and the header information text file will not be created.

```
C:\Users\IEUser>python Desktop/AudioDigger.py -i C:\Users\IEUser\Desktop\Import\Some-random-son
g.mp3 C:\Users\IEUser\Desktop\Export -head
This is not a valid mp3 file, High probability of being tempered with.
```

Figure - 41 Error message for invalid MP3 file

4.3 Spectrogram Error Handling

If you enter a invalid colour name as seen in Figure 42, AudioDigger will display an error message warning about the invalid colour name and the spectrogram will not be created.

```
C:\Users\IEUser>python Desktop/AudioDigger.py -i C:\Users\IEUser\Desktop\Import\some-random-song.
mp3 -c purple C:\Users\IEUser\Desktop\Export -sp
Color is not available. Colors are case sensitive.
```

Figure - 42 Error message for invalid colour name

If you enter a greater number for *-xmx* compared to *-xmn* as seen in Figure 43, AudioDigger will display an error message warning that the X minimum value cannot be bigger than the X maximum value. The spectrogram will also not be created.

```
C:\Users\IEUser>python Desktop/AudioDigger.py -i C:\Users\IEUser\Desktop\Import\some-random-son
g.mp3 -xmn 40 -xmx 5 C:\Users\IEUser\Desktop\Export -sp
Minimum cannot be more than the maximum.
```

Figure - 43 Error message for invalid x values

If you enter a greater number for *-ymx* compared to *-ymn* as seen in Figure 44, AudioDigger will display an error message warning that the Y minimum value cannot be bigger than the Y maximum value. The spectrogram will also not be created.

```
C:\Users\IEUser>python Desktop/AudioDigger.py -i C:\Users\IEUser\Desktop\Import\sayonara.mp3 -ymn
 7000 -ymx 1 C:\Users\IEUser\Desktop\Export -sp -s
Minimum cannot be more than the maximum.
```

Figure - 44 Error message for invalid y values

If you try to use this function using a invalid file type as seen in Figure 45, AudioDigger will display an error message warning about the invalid file format and the spectrogram will not be created.

```
C:\Users\IEUser>python Desktop/AudioDigger.py -i C:\Users\IEUser\Desktop\Import\Testfile.txt C:\U
sers\IEUser\Desktop\Export -sp
Error: Skipping  Testfile.txt . This file is unreadable. High probability of being tempered with.
```

Figure - 45 Error message for invalid file type

4.4 Hex and Bin Error Handling

If you try to use this function using a invalid file type as seen in Figure 46 and 47, AudioDigger will display an error message warning about the invalid file format and the hex/bin dump will not be created.

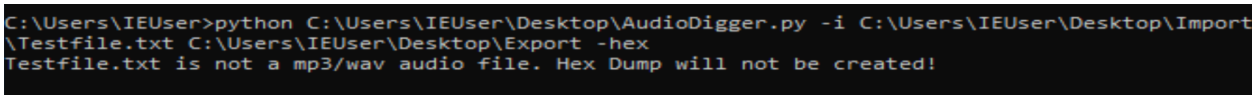

Figure - 46 Error message for invalid file type (Hex Dump)

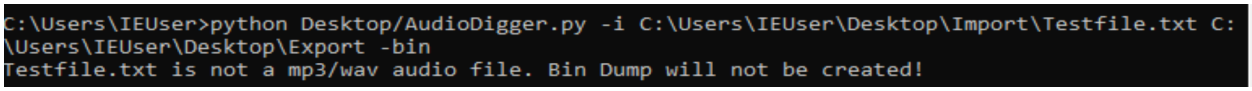

Figure - 47 Error message for invalid file type (Bin Dump)

If you enter the argument *-s* with a defined file path as seen in Figure 48 and 49, AudioDigger will display an error message warning about the invalid path and the hex/bin dump will not be created.

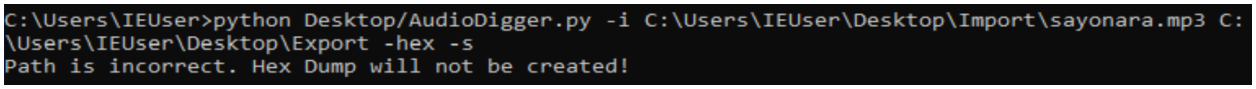

Figure - 48 Error message for using -s with defined filepath (Hex Dump)

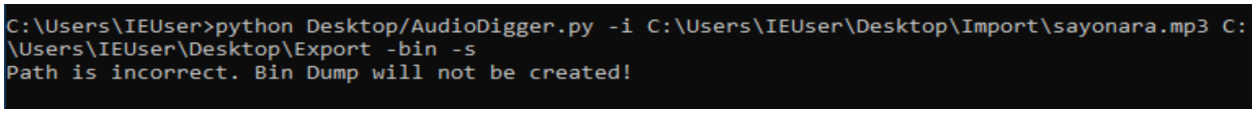

Figure - 49 Error message for using -s with defined filepath (Bin Dump)

If you try to export the header information for a directory without entering the argument *-s* as seen in Figure 50 and 51, AudioDigger will display an error message and the hex/bin dump will not be created.

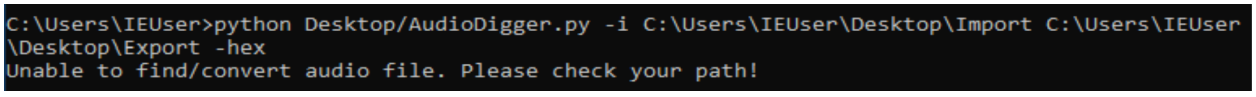

Figure - 50 Error message for invalid file type (Hex Dump)

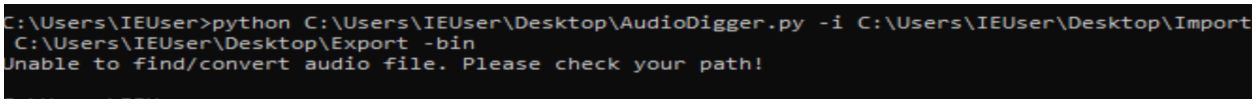

Figure - 51 Error message for invalid file type (Hex Dump)