Readme -Subtask-3 For Creating a small audio processing library

Ujjwal Mehta 2020CS10401
Avval Amil 2020CS10330

March 6, 2022

Introduction to task :- Created a library for the simulation of a pre-trained deep neural network of 4 layers in c++. **Openblas** was used for the multiplication of matrices. The activation function used was relu and softmax was used to create the final probability vector.

Explanation of Code Structure and File Structure:- In the directory which we have submitted, there is a main.cpp file which will be the driver, libaudio.so which is the generated library file, audio.h file which contains the definition of the libaudioAPI function, audio.cpp which has all the helper functions (e.g layer and topThree) and the implementation of the libaudioAPI function. Apart from these, we have included 8 files containing the pre-trained data made available (4 files for the weight matrices and 4 for the bias matrices). We also have contained the two header files mkl_matrix.h and open_blas.h which are used for multiplication using openblas. The functions are explained in detail in the comments.
The function 'libaudioAPI' takes two arguments - a string containing the filename and a three element array of the structure defined. The function will output another array of the same structure, and the elements of the arrays will have attributes which will denote the indices of the top three maximum probabilities in the softmax vector as well as the probabilities themselves, which we can use to find out which words correspond the most to the audio. The softmax vector is obtained by applying the 'layer' function 4 times which simply does the multiplication and addition of bias. Apart from the layer function, the 'relu' function is also applied 3 times between each

layer. The comments can be referred for further details.

Preprocessor Directives Used in Code:-

The directives used in our code are **vector, iostream, cmath, fstream** where vector is used to use vectors in our code, cmath for exponential function and iostream and fstream for reading and writing files. We have used two header files called 'mkl_matrix.h' and 'open_blas.h' for using multiplication using openblas.

Execution Instructions:-

Unzip the submitted folder and open the directory in the terminal, making sure that the structure of the submission is not tampered with.
Run the following command-

export LD_LIBRARY_PATH=/home/username/foo:$LD_LIBRARY_PATH

In the above command, replace "/home/username/foo" with the path of the directory containing the submitted files on your machine. After running the above command, open the 'makefile' file and in the topmost line, change the 'path' to the one on your machine. After making these changes, run the "make" command. This will create a file called 'yourcode.out' and you can run the test cases by the following command-

./yourcode.out features.txt output.txt

where 'features.txt' is the text file in the same folder as the other files of our submission (this file will have to be provided by the tester and contains the features of the audio sample (250 elements)). If not present, a file called "output.txt" will be created. If present, the probabilities and the keywords will be appended to the output.txt file. Note that the names "features.txt" and "output.txt" can be replaced by anything. You can use "make remove" to remove the yourcode.out file if necessary.
Using libaudioAPI function with Other Implementations of "main.cpp":- In order to use this function with a different "main.cpp" file than ours, first of all put your "main.cpp" file in

this directory, and in the directives make sure to put "include 'audio.h'", along with the other usual preprocessor directives. Also make sure to include the following code snippet in your "main.cpp" file -

```
typedef struct pred_t
{
int label;
float prob;
}pred_t;
```

and run the code accordingly.