International Journal of Innovative Research in Science, Engineering and Technology (IJIRSET)

| e-ISSN: 2319-8753, p-ISSN: 2320-6710| www.ijirset.com | Impact Factor: 7.512|

||Volume 9, Issue 7, July 2020||

Parallel Computing of Feature Extraction of ESC for Deep Learning

Roopesh R[1], Supriya H S[2]

P.G. Student, Department of Computer Engineering, JSS STU, Mysuru, Karnataka, India[1]

Assistant Professor, Department of Computer Engineering, Sir. MVIT College, Bangalore, Karnataka, India[2]

ABSTRACT: Most of the times, programmers don't care about how much time taken for classifying the files, they're just interested in classification accuracy and classified files. Here the analysis is done by comparing different classifiers and the result in computed but, what we believe is that along with the classification accuracy, if some interest is shown on computation speed, then our work becomes efficient and we'll obtain the result faster. This paper explains how audio classification can be done faster using parallel computing.

KEYWORDS: parallel computing, audio classification, python, feature extraction, deep learning.

I. INTRODUCTION

Parallel computing and multithreading are becoming ubiquitous for some time because of its computation speed and program efficiency. Parallel computing provides different ways to explore the programming methodology. It helps in exploiting parallelism and to overcome the problems that arises with sequential programming. The two major methods of parallel computing are multiprocessing and multithreading. As we know that multiprocessing involves multiple processes to compute wherein multithreading allows single process to have multiple threads (code segments) [1]– [3].
Audio classification is one of the major categories in supervised machine learning. In classification or in any ma-chine learning technique we mainly focus on results obtained from different machine learning techniques, we compare theresults and analyse. But, our suggestion is that along with different technique analysis we should also make sure thatour code compiles faster the work done efficiently. In order make the compilation faster we introduce the concept ofparallel computing. In this paper we are trying to parallelize environmental sound classification program which uses ESC-10 dataset for classification [4] and we used multithreading concept to compute the program parallelly [5].
This paper is divided into 7 sections which explains in detail of how parallelism is achieved.

Paper is organized as follows. Section II describes automatic text detection using morphological operations, connected component analysis and set of selection or rejection criteria. The flow diagram represents the step of the algorithm. After detection of text, how text region is filled using an In painting technique that is given in Section III. Section IV presents experimental results showing results of images tested. Finally, Section V presents conclusion.

II. MATERIAL AND METHODS

The dataset that we used in this project is ESC-10 data set and is available to download. ESC-10 dataset is a subset of ESC-50. This ESC-10 is a selection of 10 classes from the bigger dataset (i.e., ESC-50). This represents three general groups of sounds:

- transient/percussive sounds, sometimes with very meaningful temporal patterns (sneezing, dog barking, clockticking),
- sound events with strong harmonic content (crying baby, crowing rooster),
- more or less structured noise/sound scapes (rain, sea-waves, fire crackling, helicopter, chainsaw).

Each class have 50 files containing in it.
The question that arises now is why this small subset is taken for classification? Why not ESC-50 dataset? The answer to thisquestion is accuracy. This proof-of-concept dataset that was created earlier presents a different problem to tackle than thewhole ESC-50 dataset. The ESC-10 dataset results are more accurate than ESC-50. We chose ESC-10 dataset instead ofESC-50 dataset. As in [6] the author explains in detail about ESC datasets.

| e-ISSN: 2319-8753, p-ISSN: 2320-6710| www.ijirset.com | Impact Factor: 7.512|

||Volume 9, Issue 7, July 2020||

III. METHODOLOGY

In our approach we try to exploit task-based parallelism in python. Consider the following example where we can findthe task-based parallelism [7].

```
1. def analysis:
2.     data = range(20)
3.     results =[]
4. for i in data:
5.         y = analysis(i)
6.         results.append(y)
```

In our implementation also we find same kind of loop where we try to run same task for different files present in the directory. In order to simplify or parallelize this we extracted that loop and kept that in a different method so that we could put that loop into threads. we use *threading*. *py*library [8]for multithreading. this threading library allows us to create threads using threading module.
In order to implement thread using threading module the following steps are required.
Create a subclass of Thread class.
Override the*init*method to add the arguments.
Override therunmethod to add the contents of thethread.
The snippet below explains in detail on how three steps are implemented by overriding both *init* and *run* methods [9], [10].we can start the thread by implementing*start()*methodin a place where we need to run the thread.

```
1. import threading
2.
3. class myThread (threading.Thread):
4. def __init__(self, threadID, name,counter):
5.       threading.Thread.__init__(self)
6. self.threadID = threadID
7. def run(self):
8. print"Starting"+self.name
9. // codeblock - place where we can put
10.  //the method which is initialized in threads
11.  print"Exiting"+self.name
```

In our implementation, we created a method for the loop extracted. the loop which is extracted does the task of extracting a set of features (such as stft, mfcc, contrast, tonnetz, mel Specgram and chroma) for all the files that are present in the dataset. So, we put this extracted method in the *code block*of thread class to run it on different files parallelly.
Just by doing this doesn't completes multithreading task, we found different obstacles just by doing this.

- First obstacle we came across is access violation error. This error means that, all the threads that were trying to access the same resource at a time. This gives rise to deadlock situation. The solution to the is use thread synchronization mechanism. Locks are the most commonly used mechanism which is provided by threading module. we used*acquire()*to hold the resources and release()to release the resources.
- After solving the access violation error, the second obstacle we came across is race condition. The useof*join()*resolves this issue.*join()*causes the mainthread to halt until all the other threads complete itsexecution.

After resolving these two issues we were able to parallelize the sequential code. In the next section we will discuss aboutthe results obtained in detail.

| e-ISSN: 2319-8753, p-ISSN: 2320-6710| www.ijirset.com | Impact Factor: 7.512|

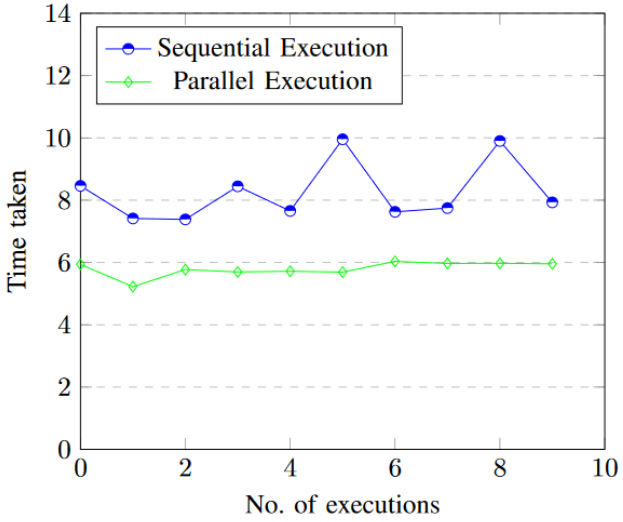
||Volume 9, Issue 7, July 2020||

IV. EXPERIMENTAL RESULTS

The following table are the details of the system andsoftware information that is used to implement this project.

Processor	AMD Ryzen 7 Quad core 2.3GHz
RAM	16GB DDR4
Graphics	6GB Nvidia Geforce GTX 1660 Max-Q
Operating System	Windows 10 64-bit
Python version	v3.8.0
IDE	JetBrains PyCharm Community Edition 2019.2.3 x64

Using the above configuration, we ran both sequential code [11] and our parallelized code [12] and computed the following graph based on time taken to complete the execution. The code is executed for all the500 files for 10 times to make sure the computational speed is achieved and the results are compared



The above graph shows the detailed comparison done between the sequential execution and parallel execution.

V. FUTURE WORK

In this paper we haven't explored multiprocessing concepts. So as a future work, we wish to explore the concepts of multiprocessing in python and then achieving parallelism through multiprocessing, and also, we can try parallelizing the different extraction methods and compute the results.

VI. CONCLUSION

The concept of parallel computing helps us to achieve max performance of the computer system given that all the constraints are taken care, as in our implementation we can clearly note that the parallelized code is faster than serialized code. This can be great asset to the programmers as it helps in achieving computation speed. It helps us to explore new ways of programming the system. Multithreading is one way of parallel computing where we can achieve parallelism but it is not limited to this alone. we can even explore more with parallel computing. As a future work multiprocessing is proposed

VII. ACKNOWLEDGEMENTS

Thanks to Mr. Karol J Piczak for explaining in detail about the ESC datasets.

I express my deep gratitude to my guide Dr.Shivamurthy P M, assistant professor at JSSSTU, Mysuru for providing continuous suggestions extended to me with immense care and zeal throughout my work.

| e-ISSN: 2319-8753, p-ISSN: 2320-6710| www.ijirset.com | Impact Factor: 7.512|

||Volume 9, Issue 7, July 2020||

REFERENCES

[1] P. S. Pacheco,An Introduction to Parallel Programming. Elsevier, 2011.
[2] J. Adams, C. Nevison, and N. Schaller, "Parallel computing to start themillennium," 03 2000, pp. 65–69.
[3] J. R. Shameem Akhter, Multi-Core Programming. Intel, 2006.
[4] K. J. Piczak, "Environmental sound classification with convolutionalneural networks," in2015 IEEE 25th International Workshop on Ma-chine Learning for Signal Processing (MLSP). IEEE, 2015, pp. 1–6.
[5] R. Brown and E. Shoop, "Modules in community: Injecting moreparallelism into computer science curricula,"SIGCSE'11 – Proceedingsof the 42nd ACM Technical Symposium on Computer Science Education,01 2011.
[6] K. Piczak, "Esc: Dataset for environmental sound classification, "inProceedings of the 23rd ACM international conferenceon Multimedia,2015,pp.1015–1018.[Online].Available:https://doi.org/10.7910/DVN/YDEPUT
[7] P.E.Hadjidoukas, R. F.Scheidegger, and A. C.Bekas, "torcpy:Supporting task parallelism in python," 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2352711020300091
[8] P. S. Foundation, "Threading - thread based parallelism." [Online]. Available: https://docs.python.org/3/library/threading
[9] G. Zaccone,Python Parallel Programming Cookbook. Packt, 2015.
[10] B. Klein, "Threadsinpython,"2020. [Online].Available: https://www.python-course.eu/threads
[11] M. Fing and Micah5, "Environmental sound classification usingdeep learningwithextractedfeatures."[Online].Available:https://github.com/mtobeiyf/audio-classification
[12] R. R. Manchanbele, "Parallel computing of featureextractionofescfordeeplearning,"2020.[Online].Available:https://github.com/RoopeshManchanbele/audio-classification-parallel