Audacity GSoC 2021 Proposal

April 13, 2021

About Me:

- **Name:** Devesh Kashyap
- **Major:** Computer Science and Engineering (CSE)
- **University:** Indian Institute of Information Technology, Ranchi, Jharkhand, India
- **Discord Username:** Kashya
- **Email:** devesh.btech.cs19@iiitranchi.ac.in
- **Languages:** English, Hindi
- **Phone:** (+91) 9667844659
- **Postal Address:** H.No.729C/11, Ashok Vihar - 1, Gurgaon, Haryana, India
- **Time Zone:** Indian Standard Time (UTC +5:30)

Coding Skills:

❖ Programming Languages

- ➢ Fluent in C/C++ with sound knowledge of OOP
- ➢ Intermediate proficiency in Python

❖ Libraries & Frameworks

- ➢ Basic ML Libraries(Numpy,Scipy,Scikit-Learn,Pandas, Seaborn, Matplotlib)
- ➢ PyTorch
- ➢ TensorFlow
- ➢ Anaconda
- ➢ wxWidgets(familiar with basics,will make myself proficient before coding period)

❖ Development Environment

- ➢ Visual Studio Code
- ➢ Vim Text Editor
- ➢ OS:Ubuntu 20.04 LTS
- ➢ HP 15,8th Gen Intel Core i5,240 GB SSD,MX110 CUDA enabled

❖ Version Control

- ➢ Git

Project: Source Separation

Abstract:

This project aims to provide a user-friendly "**Sound Source Separation**" plugin to Audacity. This plugin will split the audio input file into its constituent audio stems as per users' requirement.

The input audio track can be disassembled into the following 3 categories:

2-Stems	4-Stems	5-Stems
• Vocal • Accompaniment separations	• Vocals • Drums • Bass • Other separations	• Vocals • Drums • Bass • Piano • Other separations

The extracted stems can be further used to rebalance the audio input using Audacity's mixing desk.

The existing" **Voice Reduction and Isolation**" feature works pretty well with center-panned audio tracks, however miserably fails in all other cases. Further, this feature is too complex and not user-friendly.

The "**Sound Source Separation**" plugin aims to declutter the system and provide a user-centric experience while maintaining its integrity by utilizing State Of The Art (SOTA) Machine Learning techniques and frameworks.

Project Ideas:

- Using Spleeter as the core Sound Source Separation Software (https://github.com/deezer/spleeter)
  - Recasting Spleeter into a C++ library for easier integration
  - Using TensorFlow C API for portability
- Creating a minimalist user-friendly GUI under "**Tracks Menu**" ,with keyboard shortcuts enabled for easy access

Implementation Details:

Why Spleeter

- Spleeter was designed with ease of use, separation performance and speed in mind.
- Spleeter currently ranks 6th on MUSDB18 Benchmark.
- The ever-growing developer community of Spleeter provides extensive support.
- Spleeter is based on TensorFlow and makes it possible to-
  - Separate audio file into 2,4 or 5 stems with a singlecommand line using pre-trained models.
  - Train new source separation models or fine-tune pre-trained ones with TensorFlow (provided you have a dataset of isolated sources)
- Spleeter's pre-trained models have also been used by variousprofessional audio softwares, such as:

- ❏ iZotope in its *Music Rebalance* feature within **RX 8**
- ❏ SpectralLayers in its *Unmix* feature in **SpectralLayers 7**
- ❏ Acon Digital within **Acoustica 7**
- ❏ VirtualDJ in their stem isolation feature
- ❏ Algoriddim in their **NeuralMix** and **djayPRO** app suite

★ ***If during further discussions with the mentors, we decide to use a different splitter, or some other techniques (spectrogram based or some other), I'll revamp my entire implementation strategy, plans and timeline according to the project's requirements.***

❖ Recasting Spleeter into a C++ library

Spleeter was entirely written in Python, so for successful integration with Audacity, I had two choices -

- ➢ Making C++ bindings for python files in Spleeter and integrating through an API
- ➢ Converting Spleeter into C++ library and integrating it with TensorFlow C API

Keeping the project's time scope in mind, I chose the latter, as some work had already been done in converting Spleeter into a C++ library which could lead to faster integration with the Audacity.

The extra time acquired can be used to tackle more serious issues which may arrive after preliminary integration, like -

- **Potential memory overflow while working with large audio files:**
  - Memory Overflow in Low RAM devices is very much prevalent issue in Spleeter. During batch processing of audio clips many later processes get killed due to insufficient space. One way to counter this process flaw is to split the audio track and process partwise individually and re-combine them before presenting to the user.

- **Optimizing the models and improving the linking to decrease the separation time:**
  - There is a severe difference between the splitting time of Spleeter Models and the Import/Export time of Audacity. Furthermore, this issue is aggravated in the 5 Stem Separation model.
  - I have made a comparative report of the time taken by the Spleeter's models' to process various soundtracks of different length to the time taken by the Audacity to import/export the soundtracks.

    This report further justifies the need for optimization in Spleeter's models' for smooth and efficient integration with Audacity.

  ★ Spleeter's Performance Review for Integration with Audacity

❖ Using TensorFlow C API

TensorFlow provides a C API that can be used to build bindings for other languages. The API is defined in c_api.h and designed for simplicity and uniformity rather than convenience.

TensorFlow for C is supported on the following systems:

- Linux, 64-bit, x86
- macOS, Version 10.12.6 (Sierra) or higher
- Windows, 64-bit x86
  - ✓ Currently, Audacity also supports 32 bit systems, but the incompatibility of TensorFlow C API is a major issue, whichI'll address as a secondary goal of this project.

❑ Sample Implementation of 2-Stems Audio separation using TensorFlow C API

```
void Split(const Waveform &input, Waveform *vocals, Waveform *accompaniment,
           std::error_code &err) {
  std::vector<TFHandlePtr<TF_Tensor>> tf_output;
  RunModel(input, TwoStems, GetOutputNames(TwoStems), &tf_output, err);
  if (err) {
    return;
  }
  SetOutput(tf_output, input.cols(), {vocals, accompaniment});
}
```

The main Split function

```
std::vector<std::string> GetOutputNames(SeparationType type) {
  switch (type) {
  case TwoStems:
    return {"strided_slice_13", "strided_slice_23"};
  case FourStems:
    return {"strided_slice_13", "strided_slice_23", "strided_slice_33",
            "strided_slice_43"};
  case FiveStems:
    return {"strided_slice_18", "strided_slice_38", "strided_slice_48",
            "strided_slice_28", "strided_slice_58"};
  default:
    return {};
  }
}
```

GetOutputNames function

```
void RunModel(const Waveform &input, SeparationType separation_type,
              const std::vector<std::string>& output_names,
              std::vector<TFHandlePtr<TF_Tensor>> *result,
              std::error_code &err) {
  auto bundle = Registry::instance().Get(separation_type);
  if (!bundle) {
    err = std::make_error_code(std::errc::protocol_error);
    return;
  }
```

RunModel function

```
void SetOutput(const std::vector<TFHandlePtr<TF_Tensor>> &tf_output,
               uint64_t frame_count, std::vector<Waveform *> output) {
  for (auto index = 0; index < tf_output.size(); index++) {
    output[index]->resize(2, frame_count);
    auto tf_output_ptr = static_cast<float*>(TF_TensorData(tf_output[index]->get()));
    std::copy(tf_output_ptr, tf_output_ptr + output[index]->size(),
              output[index]->data());
  }
}
```

SetOutput function

❑ Sample GUI Implementation

Without Input Track

With Input Track

Secondary Goals/Post GSoC Goals:

❖ Plugin Support for Multiple Models

In case, I'm able to achieve the primary goal specified above beforetime,or after GSoC period,I'd like to work on providing plugin supportfor various other audio separation softwares like "**DEMUCS**", "**Tasnet**","**LaSAFT+GP'oCM**"and "**Wavenet**".

These different source separation software can be used as a downloadable add-ons. This functionality is mainly aimed towards researchers who are moreoriented towards flexibility and diversity during experiments.

❖ Providing Support for 32 bit architecture

The 32 bit architecture isn't able to support any SOTA ML modelsmainly due to incompatibility of dependencies required.

One plausible solution would be to install TensorFlow C API(or any other dependency required) from source code, compile and install it in32 bits systems,then completely modify all scripts to adapt the new environment.

But,this is quite an arduous task,and will require much greater timethan the one provided by GSoC.

Proposed Deliverables (during GSoC):

- A C++ sound separation library based on Spleeter
- Integrated minimalist GUI for regular users
- Detailed documentation; for users as well as developers
- Fortnightly blogs on development advances, milestones and problems incurred.

Milestones:

- **Milestone-1** (deliverable before coding period begins)

  I'll start by familiarizing myself with core Audacity API.Having anoverall idea of the codebase will be necessary when I go ahead creating the Spleeter C++ library.

- **Milestone-2** (deliverable before 1st evaluation)

  I'll continue to work on creating Spleeter C++ library.Once the library is provisionally created, I'll move on to setting up a CMake build system and testing if things work as expected.

  In the second half of Milestone 2, I'll start designing the GUI andwill initiate the integration process.

- **Milestone-3** (deliverable before final evaluation)

  I'll continue to work on integrating the newly created Spleeter library into Audacity,while overcoming any problems that arise.

  By the end of Milestone 3,I would have successfully integratedthe Source Separation plugin into Audacity with complete CMake scripts.

  The final software will function as a downloadable add-on in which the system will automatically download the required dependencies and core models after gaining the user's consent.

- **Milestone 4 (wish-list/if time permits)**

  In case I'm able to complete the first three milestones within time, I will work secondary goals mentioned above.

Detailed Timeline:

Present – May 17	:	Get more familiar with Audacity's Codebase & it's functionalities
May 17– June 7	:	**Community Bonding Period:** Discuss with mentors any important points missed & plan the work ahead
Coding Period Starts : Milestone 1 reached		
June 7 - June 17	:	Calibrate the existing Spleeter C++ library for Audacity's Codebase
June 18 - June 28	:	Integrate Spleeter C++ lib. with Audacity
June 29 - July 6	:	Complete the headers,setup CMake build and make sure it works as expected
July 7 - July 9	:	Extensive testing with sample audio clips & inspection of memory leaks and the header linkings
July 10 - July 12	:	Final test using new virtual env. & solving any new dependency problems
1st Evaluation : Milestone 2 reached		

July 13 - July 15	:	Solving dependency problems(if left), otherwise start preparing for GUI design
July 16 - July 26	:	Design the menu bar & check if the waveforms are generated properly and are downloadable separately without any problem
July 27 - August 7	:	Integrate the Spleeter API with GUI
August 8 - August 11	:	Setup CMake build and make sure it works as expected
Milestone 3 reached		
August 12 - August 16	:	Do a general code cleanup. Make sure there is nothing left undone and everything is tidy. Prepare for final evaluation.
Final Evaluation		
After August 16	:	Keep contributing by completing secondary goals

Why me for the Project?

I chose this particular organization because it is very well aligned with my interests and also correlated to what I have worked on in the past. Other than that, this project provides a good opportunity to apply my learnings on a practical scale. This is my field of interest and I'm naturally inclined to this Project.

I've been studying Machine Learning for a few months now. I'm fascinated by the advent of 'AI' and how it's affecting the world around us. This project seems like the perfect amalgamation of core Software Engineering and Machine Learning and I'm very much interested in contributing to it.

I have been programming for almost 3 years now. As a result, I am fairly acquainted with the specifics of C, C++, Python, OOP and Software Development Techniques. Although I'm new to WxWidgets, I can pick up the nuances of the framework fairly quickly. While doing research for this project, I came to know lot about the field of Audio Engineering and processing. This project seems like the perfect opportunity for me to deep dive into the world of Audio Engineering.

Time Contribution:

I can easily give 40-50 hours a week till my college reopens and 30-40 hrs. a week after that I intend to complete most of the work before my college reopens. Other than this project, I have no other commitments / vacations planned for the summers.

References:

- https://wiki.audacityteam.org/wiki/Developer_Guide
- https://www.aosabook.org/en/audacity.html
- https://doxy.audacityteam.org/annotated.html
- https://sigsep.github.io/
- https://www.eecs.qmul.ac.uk/~markp/2011/Evangelista11-dafx-chapter_accepted.pdf
- https://paperswithcode.com/sota/music-source-separation-on-musdb18
- https://archives.ismir.net/ismir2019/latebreaking/000036.pdf
- https://paperswithcode.com/task/music-source-separation
- https://github.com/gvne/spleeterpp

****