ECE539 Project Proposal

PROJECT TITLE: Singing livestream segmentation assistant

KEY WORDS: Audio classification, FFT, CNN, FIR, Speaker Verification

TEAM MEMBERS:

- Avi BALAM, 9086006591, abalam@wisc.edu
- YUSHANG JIN, 9083280140, yjin248@wisc.edu
- ZALISSA ZONGO KAFANDO, 9084045047, zongokafando@wisc.edu

GITHUB REPO: https://github.com/EricEricEricJin/ECE539-Group-Project/

Overview

Our target is to assist in slicing singing livestream videos, that is help cutting the singing parts out from the whole livestream recording, by highlighting the potential singing partitions. The core is audio classification that distinguish between singing and speech.

Background

"Classifying Music and Speech with Machine Learning" by Code AI demonstrated an approach to classifying audio into music or speech categories. They transfer audio into spectrogram using FFT, and use a CNN model for classification. With `gtzan` dataset (see below) the validating accuracy reaches 100%.

Statements of Work

Datasets

We will use the three data sources below, all open, free, and well labeled. One is already-preprocessed dataset and other two need collecting and preprocessing by our own (we have done it).

- `gtzan`

  gtzan_music_speech is an open source dataset with 60 speech audios and 60 music audios, each is 30 seconds long, mono channel, 22050Hz sample rate, stored in WAV format. Each audio has 30 * 22050 = 661500 samples. (i.e., shape of dataframe is (661500)).

- `xxm_splitted`

  For better performance we want to train and test the model with one same person's dataset. We obtained videos on bilibili in which a person only talk (no singing) and in which the same person only sing (no talking).

  The videos are crawled down (using Python, selenium and bilili), converted to audio, sliced into 30-second pieces, resampled at 22050KHz, and mixed into mono channel (using Python and ffmpeg). This results in the exactly same format as `gtzan` dataset.

  Below is the summary of this dataset

```
$ ls xxm_singing/*.wav | wc -l; ls xxm_speech/*.wav | wc -l
137
137

$ ffprobe -i xxm_singing/0.wav
......
Duration: 00:00:30.00, bitrate: 352 kb/s
Stream #0:0: Audio: pcm_s16le ([1][0][0][0] / 0x0001), 22050 Hz, 1 channels,
s16, 352 kb/s
```

- `xxm_mixed`

  There are also long (not splitted, ~150min each) videos in which a person (same person as in `xxm_splitted`) sometimes talks and sometimes sings. There are human-labeled timestamps of starts of each singing in the description of the video. These descriptions are crawled down, parsed using regular expression, and stored in hour,min,sec format as below (one video maps to one CSV file). They are used to test and score the performance of our final program.

  ```
  $ cat crawler/label/BV1134y1g7qX.csv
  0, 19, 11
  0, 35, 31
  1, 30, 6
  1, 35, 43
  [...trunked...]
  ```

- Crawler and data processing scripts is on Github.

Method

We have reproduced the previous work mentioned above and applied it to our use case. The procedures, outcomes, and analysis are below.

Reproduce previous work

Our Jupyter Notebook

- Data preprocessing:

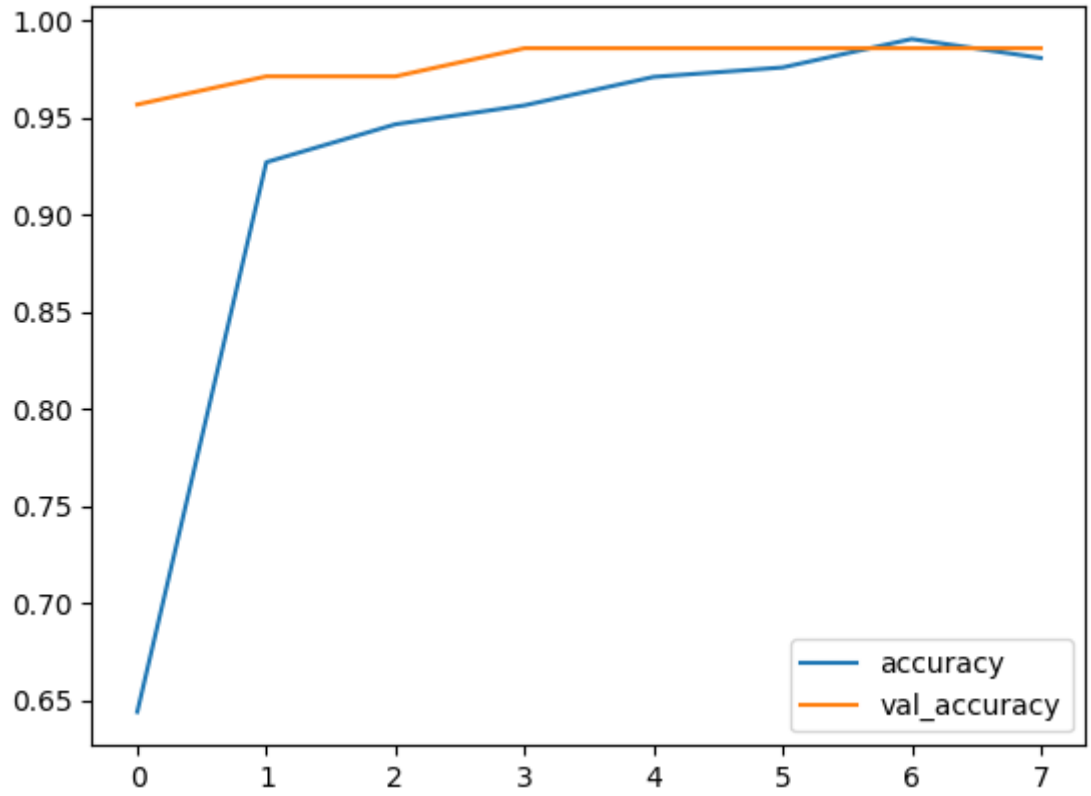
  We want the frequency domain features, so preprocessing requires transfer each 30-second audio into spectrogram (using FFT, chunk size and step size are hyper parameters to be tuned) and normalize the amplitude.

- Model:

  With above preprocessing, each input is a spectrogram matrix whose ith row is the ith chunk's frequency spectrum. Output is by one-hot coding, (e.g., music is [0 1] and speech is [1 0]). Then use CNN to connect input and output, and fit, evaluate the model.

- Result:

  Below is the validating accuracy against epoch number of the model with `xxm_splitted` dataset. We can see Code AI didn't lie and the accuracy of its model is very high.

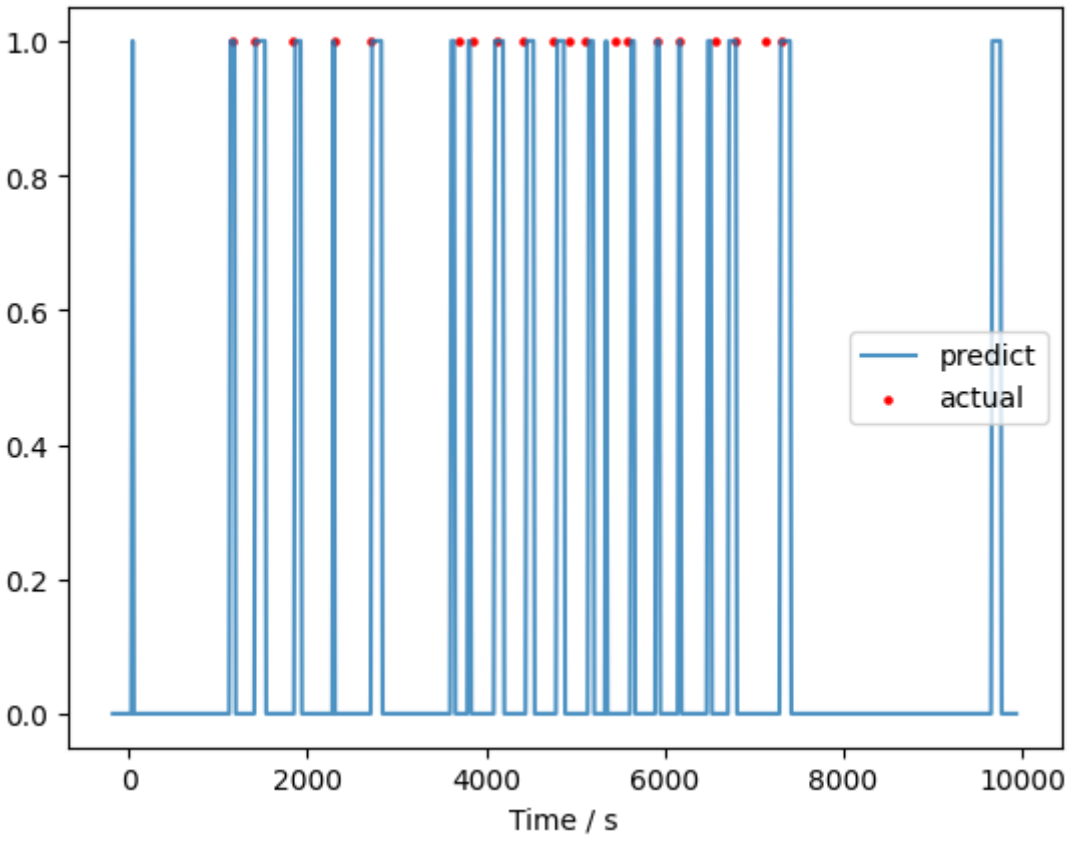


Apply previous work to our use case

Below is the block diagram of our current implementation:

```
    [Long audio]-->[CHUNK 1][CHUNK 2] ... [CHUNK N]
                        |
                        V
                  (FFT)-->[Spectrum]-->(CNN)-->[Probability of each chunk]
                                                          |
                                                          V
[Predict outcome]<--(Thredhold)<--[Filtered probability]<--(FIR)
    * The [CNN] network is trained with xxm_splitted dataset as mentioned above.
```

Below is the outcome, blue line is the prediction (1 for singing, 0 for speech) and red dot is actual label showing starting of singing.



We (human) listened to the false positive points and find at those points there is background music making the model predict them as singing. To fix this issue, speaker recognition may be applied to recognize the specific person's voice print.

Also, the FIR filter I applied is used to detect rising edges, it will filter out long continuous positive points. This is consistent with our goal since we assume each song is in certain length and there is no continuous singings.

Challenges and what to do next

- The 'threshold' should not even exist

  As shown in above model block diagram, a chunk will be classified as singing if the probability of singing is higher than threshold. Theoretically there shouldn't be a threshold, as the network is trained without it. But with a few tests already done we noticed the FPR is sometimes too high if no threshold (i.e., use output of the CNN directly), even with a filter.

  We will find out why this happens, and perhaps expanding the model (add more layers) can help solve this issue? This issue may also be relavent to the noise (see below).

- Noise: neither singing nor speech

  Each audio in our training data `xxm_splitted` and `gtzan` dataset is either singing or speech, but in the whole mixed audio `xxm_mixed` there are chunks that are neither singing nor speech, instead they are noise (sometimes mixed with low amplitude background music).

  We need to find a way to split these noise inputs away from singing and speech.

  Instead of adding another class (turn binary classification into a 3-class classification), we may try to recognize noise at preprocessing stage, as they always have lower amplitude. But how low? The

threshold need to be determined and may be trained as a parameter using regression?

- False alarmed BGMs

  Sometimes there are BGMs singing by other singers inside the mixed audio which will be predicted as false positive. We may apply Vocal Isolation and Speaker Verification to predict whether this singing is by the livestreamer or others. They may increase specificity but will definatily decrease TPR.

  For our case high TPR is more important than high specificity (explained below). We will have a try and check the confusion matrices to determine whether we should have apply the two models.

Outcome and evaluation

As mentioned, the labels in `xxm_mixed` are start timestamps of each singing, and model output are probabilities that each chunk is singing. By plotting them together as above figure, it is easy for human to see if they are consistent or not. But we must find a convincing way to mathematically evaluate the accuracy.

Also, as this program targets to navigate human to potential singing timestamps, it is acceptable if there are small time shifts between actuals and predictions. We need to determine how small is considered acceptable. The time shift is caused by convolution with FIR filter, can be mathematically offsetted.

Due to the target of our program, we want a high sensitivity (TP/P), and can sacrifice false alarm rate, because human can quickly go through false alarmed points and find it is actually not singing, but false negative will lead to missing of some parts. We will stress this when developing and evaluating our model.

Project Plan

- *Initial Project Proposal* :

  The project begins with the preparation of an initial project proposal outlining the goals, methods, and expected outcomes. This proposal serves as a roadmap for the entire project. Tentative Completion Date: (10/20/2023)

- *Environment Setup* :

  We will build up our project environment utilizing Google Colab and GitHub for easy collaboration and development. The GitHub repository will include all project-related documents, including code, figures, diagrams, and the final presentation slides.

- *Verify Previous Works Model* :

  We go through the previous works model as seen in this proposal already, we will be going even in depth to ensure critical analysis and understanding feasibility of the project, and the usage of Tensorflow.

- *Research Optimiazation* :

  To enhance our audio classification model, we will conduct a comprehensive review of existing models in the field of online videos in our chosen platform. Our goal is to identify potential optimizations that can be applied to our model, leading to improved accuracy and efficiency.

- *Data Collecting and Preprocessing* :

Data preprocessing is a critical step in preparing the dataset for training. We need to clean and preprocess the data, ensuring that they are appropriately formatted and ready for use in our classification model.

We have collected and preprocessed the data from designated sources, as seen above.

- *Begin Convolutional Neural Network Model* :

  To optimized existing models, we will dive into CNNs, understand how they work, and build our own models in TensorFlow.

- *Submit Project Progress Report* :

  Summarize the work done so far, including data preprocessing, the binary classification model's development, and any initial findings or challenges encountered. Tentative Completion Date: (11/18/2023)

- *Train CNN Model* :

  Continuing our work, we will train the CNN model using the preprocessed data. Simultaneously, we will commence the development of data data-adaptive training algorithm as well.

- *Test Model* :

  We will rigorously test our CNN model on the testing dataset to evaluate its accuracy and performance. This process involves feeding untrained images from the same distribution to the model and assessing its predictive capabilities. The results will be logged and used to determine the most effective model.

- *Evaluate Results* :

  An in-depth evaluation of the model's performance will be carried out, including metrics such as accuracy, precision, recall, and F1 score. The goal is to assess how well the model classifies the different audio segments and by how much degree it would be useful in a rela life application.

- *Visualize Results* :

  Visualization of the model's performance and classification results will be essential to convey our findings effectively. Visual aids, such as plots and charts, will be used to present the results clearly and intuitively. This will heavily use the matplotlib libraries of Python.

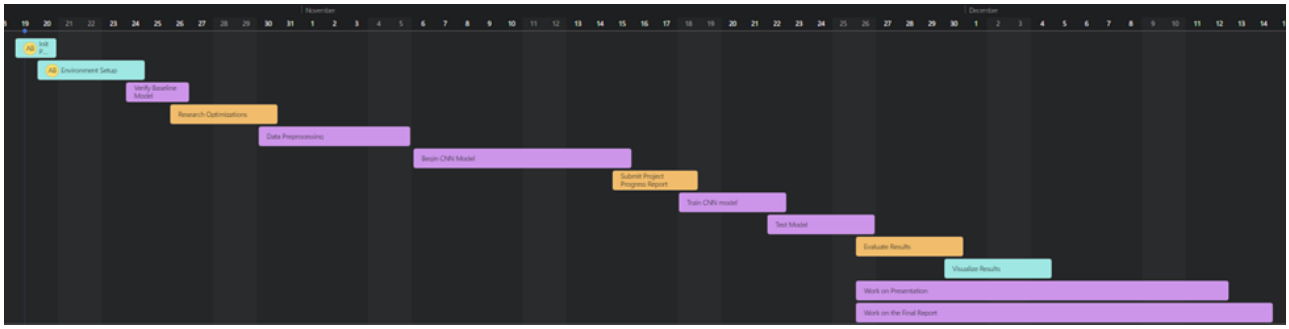
- *Work on Presentation* :

  The project presentation will be prepared, summarizing the entire project, methodologies, findings, and key takeaways. This presentation is crucial for sharing our work with peers and stakeholders. Tentative Completion Date: (12/11/2023)

- *Work on the Final Report* :

  The final project report will be composed, providing a comprehensive overview of the entire project. It will include details on the problem statement, methodology, results, discussion, and conclusions. The report serves as a record of our work and findings. Tentative Completion Date: (12/14/2023)

- *Gantt Project Timeline* :



Frameworks and computing recourses

- Tensorflow and Keras
- Tesla T4 on Google Colab

About some concerns raised by Professor

- Computation time

  Upon our reproduction of previous work, we find the preprocessing (FFT), training, and testing is fast on T4 GPU.

- Dataset size

  Professor recommends us to extract the features from audio and only save the features as dataset. However, in our case the feature is spectrogram. Raw audio is in size (661500,) and spectrogram is currently in size (5166, 129) (depends on FFT sliding window size and step), which is even larger.

  The GBs dataset may be difficult to distribute on Github, but they can be handled by google drive and Colab easily.

References

- Code AI Blogs "Classifying Music and Speech with Machine Learning." Medium, https://medium.com/m2mtechconnect/classifying-music-and-speech-with-machine-learning-e036ffab002e, 2021.
- "GTZAN music/speech collection", Kaggle, https://www.kaggle.com/datasets/lnicalo/gtzan-musicspeech-collection/