MusicList

Music Playlists Download from Youtube API

By Banana 9 fingers

MEMBERS

6688010
Pakkapol Boonluck

6688046
Warut Khamkaveephart

6688050
Chaiwat Kor-u-thaisathian

6688083
Teeramanop pinsupa

6688087
Thanakorn Praditkanok

ISSUE

First, we try to make a visualization on audio using **Deep Music Visualizer** project from https://github.com/msieg/deep-music-visualizer.

But the project is established **6 years ago.**

So we cannot apply it because some modules, version of library are not fitted to present program.

We think another project to create instead.

OVERVIEW

This project involves building a YouTube video downloader using the YouTube Data API and yt-dlp, designed to run in Google Colab. The downloader retrieves video metadata and downloads content, with the added objective of analyzing performance across various execution strategies and hardware accelerators.

OBJECTIVE

- Download videos from YouTube using the YouTube API and yt-dlp.
- Implement and compare three execution models:
  - Sequential processing
  - Multithreading (concurrent.futures)
  - Multiprocessing (multiprocessing)
- Benchmark and compare execution speed across three different hardware backends:
  - CPU
  - GPU
  - TPU

PROBLEM

1. **Rate Limiting by YouTube** (HTTP 429: Too Many Requests)
Sending too many requests in a short period—especially in loops or parallel processes—can trigger YouTube's rate-limiting mechanism, resulting in an HTTP 429 error. This temporarily blocks access and disrupts the downloading process.

2. **Authentication and Cookie Requirements**
Some YouTube videos (e.g. age-restricted or region-locked content) require user authentication. Without supplying valid browser cookies to yt-dlp, these videos may not be downloadable, limiting access to the full range of content.

3. **Colab Resource Usage Restrictions**
Running too many operations or making repeated requests in a single Colab session can lead to usage restrictions, including:

- Session disconnects
- Temporary suspension from accessing GPU/TPU
- Slower performance due to throttling

ENVIRONMENT

```python
def search_youtube_by_mood(mood):
    try:
        youtube = build(YOUTUBE_API_SERVICE_NAME, YOUTUBE_API_VERSION, developerKey=API_KEY)
        search_response = youtube.search().list(
            q=mood,
            part='id,snippet',
            order='relevance',
            type='video',
            videoDuration='short',
            maxResults=numMusic
        ).execute()
    except HttpError as e:
        print(f"An HTTP error occurred: {e}")
        return []

    return [{
        'videoId': item['id']['videoId'],
        'title': item['snippet']['title']
    } for item in search_response.get('items', [])]
```

- q - mood
- part - id, snippet
- order - relevance
- type - video
- videoDuration - short
- maxResults - xx

- **mood** - input mood
- **id, snippet** - video detail
- **relevance** - sorted based
- **video** - request type
- **short** - less than 4 min
- **xx** - no. of request

CODE REVIEW

Sequential processing

```
# Process all videos (no multithreading)
def process_playlist(mood):
    videos = search_youtube_by_mood(mood)
    results = []
    for v in videos:
        file = download_video(v['videoId'], v['title'])
        results.append(file)
    return results
```

CODE REVIEW

Threading (ThreadPoolExecutor)

```python
# Process playlist using multi-threading
def process_playlist(mood):
    videos = search_youtube_by_mood(mood)
    if not videos:
        return []

    with concurrent.futures.ThreadPoolExecutor() as executor:
        downloaded_files = list(executor.map(lambda v: download_video(v['videoId'], v['title']), videos))

    return downloaded_files
```

CODE REVIEW

Multiprocessing (ProcessPoolExecutor)

```python
# Process playlist using multiprocessing
def process_playlist(mood):
    videos = search_youtube_by_mood(mood)
    if not videos:
        return []

    with concurrent.futures.ProcessPoolExecutor(max_workers=multiprocessing.cpu_count()) as executor:
        downloaded_files = list(executor.map(download_video, videos))

    return downloaded_files
```

OUTPUT

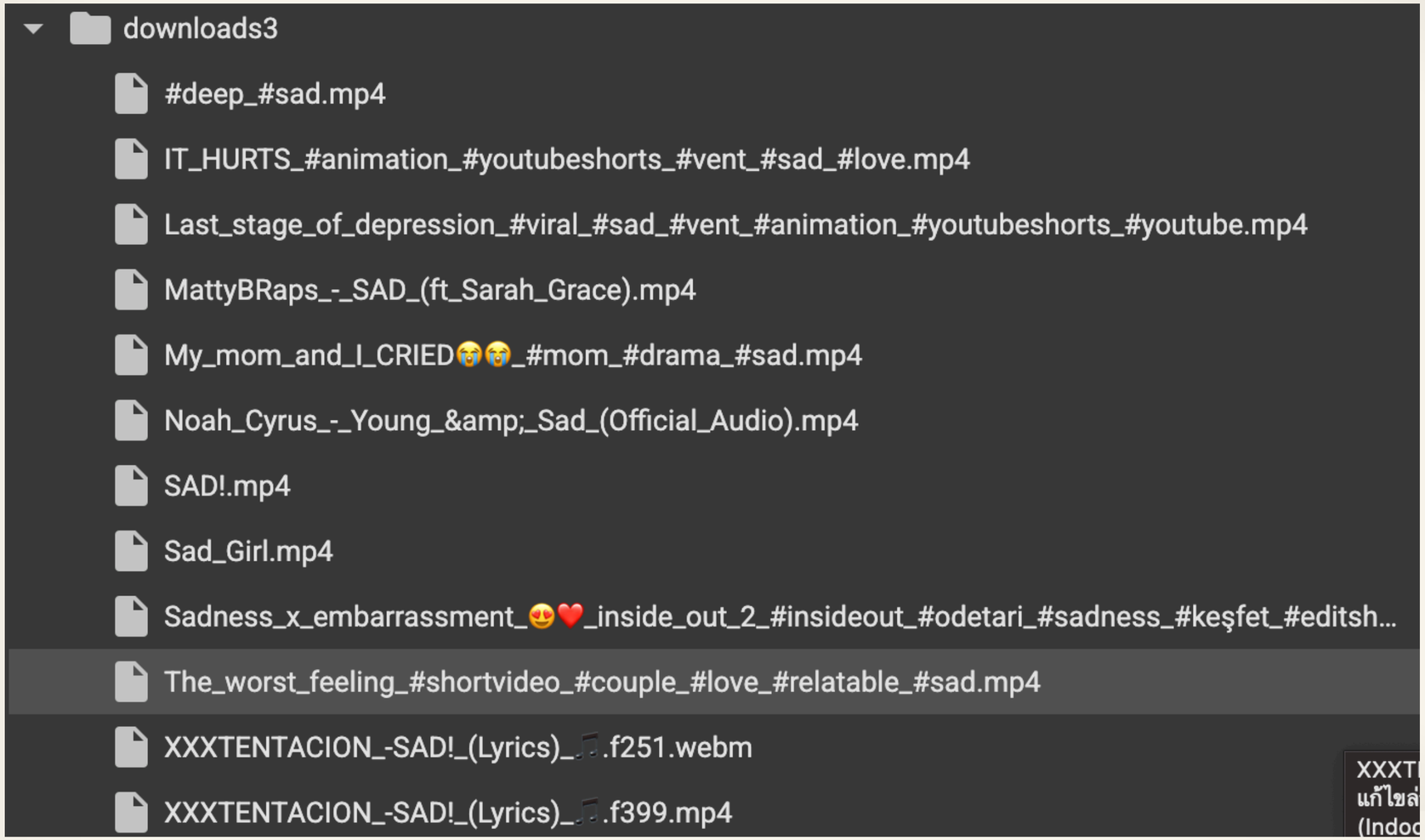
```
Enter the mood for your playlist: sad
Processing playlist for mood: sad
Downloading: XXXTENTACION - SAD!
Downloading: XXXTENTACION - SAD! (Music Video)
Downloading: SAD!
Downloading: Noah Cyrus - Young &amp;amp; Sad (Official Audio)
Downloading: The worst feeling #shortvideo #couple #love #relatable #sad
Downloading: MattyBRaps - SAD (ft Sarah Grace)
Downloading: Last stage of depression #viral #sad #vent #animation #youtubeshorts #youtube
Downloading: sad songs for when your past hurts 💔
Downloading: Sad guitar beats
Downloading: IT HURTS #animation #youtubeshorts #vent #sad #love
Downloading: Sadness x embarrassment 😍❤️ inside out 2 #insideout #odetari #sadness #keşfet #editshort #emotions
Downloading: #deep #sad
Downloading: My mom and I CRIED😭😭 #mom #drama #sad
Downloading: True Hero&amp;#39;s Are On The Inside #heroic #sad
Downloading: The worst kind of sad is not being able to EXPLAIN why 🙃 #shorts #quotes
Downloading: XXXTENTACION -SAD! (Lyrics) 🎵
Downloading: XXXTENTACION - SAD (Clean Version)
Downloading: Hurt people hurt people. #toystory #toystory3 #lotaso #sadcartoon #sad #sadscene #fypシviral #fypシ
Downloading: sanah - Sad
Downloading: ❗SAD❗ Moments that make you cry 🥹 #shorts #shortsfeed
```

```
Downloaded files:
 - downloads1/XXXTENTACION_-_SAD!.mp4
 - downloads1/XXXTENTACION_-_SAD!_(Music_Video).mp4
 - downloads1/SAD!.mp4
 - downloads1/Noah_Cyrus_-_Young_&amp;amp;_Sad_(Official_Audio).mp4
 - downloads1/The_worst_feeling_#shortvideo_#couple_#love_#relatable_#sad.mp4
 - downloads1/MattyBRaps_-_SAD_(ft_Sarah_Grace).mp4
 - downloads1/Last_stage_of_depression_#viral_#sad_#vent_#animation_#youtubeshorts_#youtube.mp4
 - downloads1/sad_songs_for_when_your_past_hurts_💔.mp4
 - downloads1/Sad_guitar_beats.mp4
 - downloads1/IT_HURTS_#animation_#youtubeshorts_#vent_#sad_#love.mp4
 - downloads1/Sadness_x_embarrassment_😍❤️_inside_out_2_#insideout_#odetari_#sadness_#keşfet_#editshort_#emotions.
 - downloads1/#deep_#sad.mp4
 - downloads1/My_mom_and_I_CRIED😭😭_#mom_#drama_#sad.mp4
 - downloads1/True_Hero&amp;#39;s_Are_On_The_Inside_#heroic_#sad.mp4
 - downloads1/The_worst_kind_of_sad_is_not_being_able_to_EXPLAIN_why_🙃_#shorts_#quotes.mp4
 - downloads1/XXXTENTACION_-SAD!_(Lyrics)_🎵.mp4
 - downloads1/XXXTENTACION_-_SAD_(Clean_Version).mp4
 - downloads1/Hurt_people_hurt_people._#toystory_#toystory3_#lotaso_#sadcartoon_#sad_#sadscene_#fypシviral_#fypシ.
 - downloads1/sanah_-_Sad.mp4
 - downloads1/❗SAD❗_Moments_that_make_you_cry_🥹_#shorts_#shortsfeed.mp4
```

Measure performance part

```
Execution Time: 171.87 seconds
Memory Usage: Initial - 124.74 MB | Final - 189.47 MB
Total memory usage: 64.73 MB
```

RESULT


downloads3
#deep_#sad.mp4
IT_HURTS_#animation_#youtubeshorts_#vent_#sad_#love.mp4
Last_stage_of_depression_#viral_#sad_#vent_#animation_#youtubeshorts_#youtube.mp4
MattyBRaps_-_SAD_(ft_Sarah_Grace).mp4
My_mom_and_I_CRIED😭😭_#mom_#drama_#sad.mp4
Noah_Cyrus_-_Young_&_Sad_(Official_Audio).mp4
SAD!.mp4
Sad_Girl.mp4
Sadness_x_embarrassment_😍❤️_inside_out_2_#insideout_#odetari_#sadness_#keşfet_#editsh...
The_worst_feeling_#shortvideo_#couple_#love_#relatable_#sad.mp4
XXXTENTACION_-SAD!_(Lyrics)_🎵.f251.webm
XXXTENTACION_-SAD!_(Lyrics)_🎵.f399.mp4

COMPARISON

CPU

Mood (No. of Request)	Sequential	Thread	Multiprocessing
sad (20)	171.87	78.22	84.62
sad (50)	360.4	203.01	242.95
happy (20)	158.09	95.32	117.73
happy (50)	455.7	301.21	329.92

unit: sec.

COMPARISON

GPU

Mood (No. of Request)	Sequential	Thread	Multiprocessing
sad (20)	95.3	59.95	77.82
sad (50)	282.79	194.23	210.39
happy (20)	133.32	83.21	83.21
happy (50)	376.52	269.71	285.81

unit: sec.

COMPARISON

TPU

Mood (No. of Request)	Sequential	Thread	Multiprocessing
sad (20)	86.45	52.12	30.23
sad (50)	240.66	128.02	93.96
happy (20)	103.62	59.52	32.06
happy (50)	276.2	130.21	91.03

unit: sec.

CONCLUSION

Sequential Processing

🔧 How it works:

- Downloads videos one by one in a loop.
- Only one task is running at any time.

⏱️ Performance:

- Slowest of the three.
- Total time = sum of time for all downloads.

📦 Why it's slower:

- It doesn’t use available CPU cores or threads to do tasks in parallel.
- It waits for one download to finish before starting the ne

CONCLUSION

Multithreading

🔧 How it works:

- Uses multiple threads to download videos concurrently.
- Threads share the same memory space.

⏱️ Performance:

- Much faster than sequential for I/O-bound tasks like downloading or using APIs.
- Can download multiple videos or fetch metadata at the same time.

📦 Why it's faster:

- Downloading is I/O-bound (waiting on the network), not CPU-heavy.
- While one thread is waiting for a server response, other threads can continue downloading or processing.
- Ideal for tasks like:
  - Downloading YouTube videos using yt-dlp
  - Making multiple HTTP API calls (like YouTube Data API)
  - Simulating multiple fetch() calls in Python

CONCLUSION

Multiprocessing

🔧 How it works:

- Launches separate processes with their own Python interpreter and memory space.

⏱️ Performance:

- Faster than sequential, but often not faster than threading for I/O tasks like downloading.
- Can be slower than threading due to process creation and communication overhead.

📦 Why it’s not the best here:

- Best for CPU-bound tasks (e.g., video processing, ML inference).
- I/O-bound tasks don’t benefit as much, and starting/stopping processes takes time.
- Higher memory usage and slower communication between processes.

CONCLUSION

Method	Best For	Speed (in this project)	Notes
Sequential	Simplicity	🐢 Slow	Easy but inefficient for many downloads
Multithreading	I/O-bound tasks	🚀 Fastest	Perfect for downloads and API calls
Multiprocessing	CPU-bound tasks	⚙️ Moderate	Great for heavy computation, not I/O

CONCLUSION

Hardware	Your Project Result	Why It Performs This Way
TPU	🏆 Fastest	Colab TPUs may offer faster backend I/O handling, lower VM latency, or high-performance runtime environments.
GPU	⚡ Faster than CPU	Google's GPU runtimes often come with optimized VMs, faster networking, and better parallel handling of I/O + compute tasks.
CPU	⚙️ Still good	Handles threading well, but typically has lower parallel efficiency and shared resource limits in Colab.

CONCLUSION

- TPU runtimes in Colab might not run Python directly on the TPU, but still provide high-performance VMs that can optimize I/O.
- GPU runtimes often get better infrastructure with more RAM, faster disk/network, and less resource contention.
- CPU runtimes works fine, but it's limited by core count and shared resources in free-tier environments.

REAL-WORLD APPLICATION

1.Automated Video Archiving
Download and archive YouTube content for offline use or data analysis.

2. Custom Video Analytics
Analyze video metadata (views, comments) for market research or content strategies.

3. Video Processing Pipeline
Transcode and optimize videos for social media or e-learning platforms.

4. Content Recommendation
Build recommendation engines based on downloaded video data for personalized user experiences

THANK YOU