VisualTunes

Music Playlists with Deep Learning visuals

By Banana 9 fingers

PLAN

- Create Youtube API key
- Implement 4 python coding
  - normal - without any OS concept applied
  - multi-threading applied - compare **time taken**
  - memory management applied - compare **memory usage**
  - combine many concepts (multi-threading, multi-processing, memory allocation) - compare **time taken, memory usage, CPU usage**
- Compare each coding
- Conclude in the table

DATA COLLECTED

- Youtube API key - with trying to implement - **AIzaSyBTEzNV2WvpTxaD8ZzuEDogkaWra*******
- Deep Music Visualizer (outsource project) - **https://github.com/msieg/deep-music-visualizer**
- yt-dlp - **https://github.com/yt-dlp/yt-dlp**

DONE

- Implemented 4 coding program – **NO run yet**

https://colab.research.google.com/drive/1R8DoytxG7aeHQSjwIVCCRufrSt1oXf_k?usp=sharing

- Created report - **OS concepts used and why**

```
    try:
        subprocess.run(cmd, check=True)  # Runs as a separate process
    except subprocess.CalledProcessError as e:
        print(f"Visualization failed: {e}")
        return None
    return output_file
```

```
# Use ThreadPoolExecutor for downloading videos in parallel
with concurrent.futures.ThreadPoolExecutor() as executor:
    audio_files = list(executor.map(lambda v: download_audio(v['videoId'], v['title']), videos))
```

```
    print(f"Combining {audio_file} with visual {visual_file} into {final_output}")
    # Simulating combining files (replace with actual FFmpeg logic)
    time.sleep(1)  # Simulate combining time
    print(f"Final video created: {final_output}")
```

```python
# Memory tracking function
def get_memory_usage():
    process = psutil.Process()
    return process.memory_info().rss / (1024 * 1024)  # Convert bytes to MB
```

```python
def manage_memory():
    print("Managing memory...")
    gc.collect()  # Force garbage collection to release unreferenced memory
    print(f"Memory after garbage collection: {psutil.Process().memory_info().rss / (1024 * 1024):.2f} MB")
```

```python
def measure_performance_os_applied(mood):
    start_time = time.time()  # Start time
    process = psutil.Process()
    initial_memory = process.memory_info().rss / (1024 * 1024)  # in MB

    print(f"Starting OS-applied code for mood: {mood}")
    final_videos = process_playlist(mood)

    final_memory = process.memory_info().rss / (1024 * 1024)  # in MB
    end_time = time.time()  # End time

    print(f"OS-applied Code Execution Time: {end_time - start_time:.2f} seconds")
    print(f"OS-applied Code Memory Usage: Initial - {initial_memory:.2f} MB, Final - {final_memory:.2f} MB")
```

REAL-WORLD APPLIED

1.) Personalized Music Playlists - Your app generates playlists based on mood. This is helpful for:

- Users who want to match music to emotions (e.g., relaxing, happy, focus).
- Daily use like studying, working out, or meditating.

2,) Mental Health & Wellness Apps - Music is often used in therapy or wellness apps. This project can:

- Be integrated into mental health platforms to suggest calming or uplifting music.
- Help people feel better through customized playlists with emotional support visuals.

THANK YOU