Multimedia Tools and Applications
https://doi.org/10.1007/s11042-020-10236-6

1171: REAL-TIME 2D/3D IMAGE PROCESSING WITH DEEP LEARNING

Client-driven animated GIF generation framework using an acoustic feature

Ghulam Mujtaba[1] · Sangsoon Lee[1] · Jaehyoun Kim[2] · Eun-Seok Ryu[2]

Received: 4 July 2020 / Revised: 23 September 2020 / Accepted: 9 December 2020 /
Published online: 12 February 2021

Abstract

This paper proposes a novel, lightweight method to generate animated graphical interchange format images (GIFs) using the computational resources of a client device. The method analyzes an acoustic feature from the climax section of an audio file to estimate the timestamp corresponding to the maximum pitch. Further, it processes a small video segment to generate the GIF instead of processing the entire video. This makes the proposed method computationally efficient, unlike baseline approaches that use entire videos to create GIFs. The proposed method retrieves and uses the audio file and video segment so that communication and storage efficiencies are improved in the GIF generation process. Experiments on a set of 16 videos show that the proposed approach is 3.76 times more computationally efficient than a baseline method on an Nvidia Jetson TX2. Additionally, in a qualitative evaluation, the GIFs generated using the proposed method received higher overall ratings compared to those generated by the baseline method. To the best of our knowledge, this is the first technique that uses an acoustic feature in the GIF generation process.

Keywords Animated GIF · Acoustic feature · Client-driven

Jaehyun Kim and Eun-Seok Ryu are contributed equally.

✉ Jaehyoun Kim
jaekim@skku.edu

✉ Eun-Seok Ryu
esryu@skku.edu

Ghulam Mujtaba
mujtaba@gachon.ac.kr

Sangsoon Lee
sslee@gachon.ac.kr

[1] Department of Computer Engineering, Gachon University, Seongnam, Korea

[2] Department of Computer Education, Sungkyunkwan University (SKKU), Seoul, Korea

1 Introduction

In this technological era, accessibility and sharing of multimedia content on social media platforms have increased as the speed and reliability of internet connections have improved. Animated images, such as graphical interchange format images (GIFs), are exceedingly popular; notably, they are used to share varied kinds of stories, summarize events, express emotion, gain attention, and enhance (or even replace) text-based communication [2]. There are many versatile types of media formats, but GIFs have become prevalent over the last decade owing to their distinct and unique features, such as instantaneous (very short in duration) and visual storytelling (no audio involved). Owing to their low bandwidth requirement and lightweight nature, GIFs are also integrated into streaming platforms to highlight videos. Figure 1 shows an example of animated images (WebP) on YouTube that are employed to instantaneously highlight a recommended video. Notably, there have only been a few generation studies for GIFs in multimedia research, despite their increasing popularity and unique visual characteristics.

According to a recent study [31], more than 500 million users spend approximately 11 million hours every day on the GIPHY website watching GIFs. Nevertheless, no real-time, lightweight GIF generation framework has been established, particularly for streaming platforms, despite the ubiquitous adoption and prevalence of GIFs. Server-driven techniques can provide real-time solutions to this problem, as all the information, including user data and video content, already exists on servers. There are three main concerns regarding server-driven solutions: (i) provision of real-time response to many concurrent users with limited computational resources; (ii) user privacy violation in a personalized approach; and (iii) the fact that current solutions process entire videos to create GIFs, which increases the overall computation time and demands substantial computational resources. These are the key factors that prompted us to research a conversational method and to explore a lightweight client-driven technique for GIF generation.

This paper proposes a novel, lightweight, and computationally efficient client-driven framework that requires minimal computational resources to generate animated GIFs. It analyzes an acoustic feature to track and estimate the timestamp corresponding to the maximum pitch (henceforth referred to as the "maximum pitch timestamp") from the climax

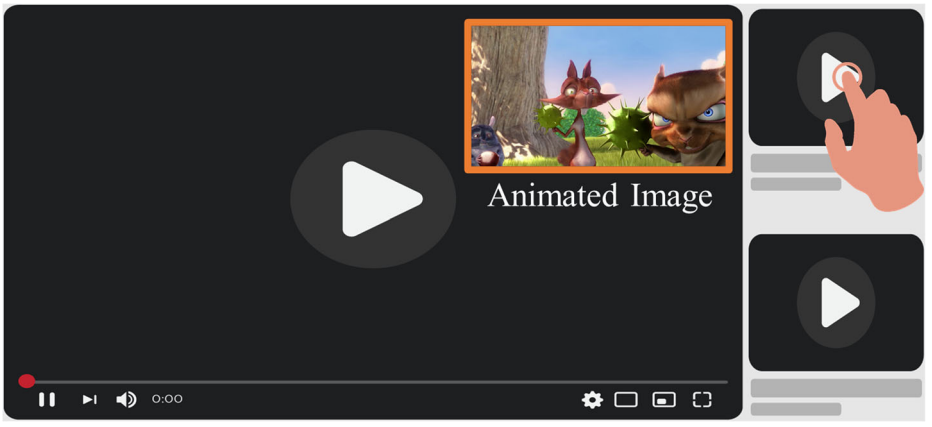

Fig. 1 Streaming platforms commonly use animated images to highlight recommended videos

section of the corresponding video. Instead of processing the entire video, it processes a small segment of the video to generate a GIF. This makes the process efficient in terms of computational resources, communication, and storage. Sixteen publicly broadcast videos are analyzed to evaluate the effectiveness of the proposed approach.[1] The main innovative technical contributions of this study are summarized as follows:

- A novel lightweight client-driven method is proposed to generate animated GIFs in accordance with user preferences.
- A two-dimensional convolutional neural network (CNN) model is designed to identify audio files according to the interest of the user..
- To validate the effectiveness of the proposed framework, Hypertext Transfer Protocol (HTTP) Live Streaming (HLS) clients and servers are locally configured.
- Extensive quantitative experimental results obtained for 16 videos show that the proposed method is 3.76 times more computationally efficient than the baseline when an Nvidia Jetson TX2 is employed as the client device. Moreover, it is 1.87 times more computationally efficient than the baseline when a device with substantial computational resources is employed (details in Section 4.3). Qualitative evaluations were conducted on 16 videos with the collaboration of 16 participants (details in Section 4.4)

To the best of our knowledge, this is the first attempt to design an entirely client-driven technique to generate animated GIFs using an acoustic feature for streaming platforms.[2]

The remainder of this paper is organized as follows. Section 2 briefly describes a summary of related work. Section 3 presents the details of the proposed client-driven framework. Section 4 presents the qualitative and quantitative results, along with the discussion. Finally, the concluding remarks are provided in Section 5.

2 Related research

This section briefly reviews the related research on animated GIF generation methods. We also review current music genre classification (MGC) methods; notably, MGC is an important technique for classifying audio files based on the genre preference of users in the proposed method.

2.1 GIF generation methods

In recent years, there has been a growing interest in researching animated GIFs. Many qualities of animated GIFs that make them more engaging than videos and other media on social network websites have been identified [2]. Facial expressions, histograms, and aesthetic features have been predicted and compared [18] to determine the most suitable video features that express useful emotions in GIFs. Another recent study [22] used sentiment analysis to estimate the textual-visual sentiment score for annotated GIFs. Several researchers have

[1] Here, we mainly focus on music videos that have a plot structure. Such music videos increase the attention of the user and are more popular than ones that do not have a plot [40].

[2] Our code and trained models are publicly available at https://github.com/iamgmujtaba/gif-acoustic.

collected and prepared datasets to annotate animated GIFs [14]. Particularly, they have collected the Video2GIF dataset for highlighting videos and extended it to include emotion recognition [13]. The GIFGIF+ dataset has been proposed for emotion recognition [4]. Another dataset, Image2GIF, has been proposed for video prediction [46], together with a method to generate cinemagraphs from a single image by predicting future frames.

2.2 MGC methods

MGC has become a prevalent topic in machine learning since a seminal report [38] was published. MGC has commercial value, but in addition, it has many practical applications such as music recommendation [39], music tagging [6], and genre classification [5]. Recent research [8, 41] has shown that spectrograms, such as short-term Fourier transform spectrograms and Mel-frequency cepstral coefficient (MFCC) spectrograms, transformed from audio signals, can be successfully applied to MGC tasks. This is owing to their capability of describing temporal changes in energy distributions with respect to frequency. CNN models have been used for different MGC tasks. In early studies on MGC using neural networks [37], researchers confirmed that techniques such as dropout, use of rectified linear units, and Hessian-free optimization can enhance feature learning effects. To exploit the feature learning capability of CNNs, an initially trained CNN was used as a feature extractor and then the extracted feature was used as a classifier [20]. The researchers achieved good results on the GTZAN [38] dataset by combining extracted features with the majority voting method. The CNN-based approaches obtain notable results in MGC tasks; however, they neglect spectrogram temporal information, which may be useful. Based on this reasoning, the long short-term memory recurrent neural network (RNN) has been used [9] to extract features from scatter spectrograms [1] of audio segments and fuse them with those obtained using CNNs. In addition, to take advantage of both CNNs and RNNs, a convolutional RNN has been designed for music tagging [7] . By adopting this, both the spatial and temporal information of the spectrograms is used.

Despite the extensive research on GIFs and MGC, a lightweight client-driven GIF generation technique specialized for streaming platforms has not been developed. Most modern end-user devices have low computational resources, and analyzing entire videos to generate animated GIFs is time consuming. This is not feasible for real-time solutions. This paper presents a novel method to generate animated GIFs on end-user devices. It uses an acoustic feature and video segments, which makes it computationally efficient and robust enough to create GIFs in real-time. The following section explains the major components of the proposed framework and GIF generation process using an acoustic feature.

3 Proposed framework

Streaming platforms manage video and audio separately for each video. The video is split and stored in small continuous segments. Dividing a video into segments and separating audio allows the streaming platforms to manage them separately according to different specifications. As described in Section 2, existing techniques process the entire video to generate an animated GIF, which is not an efficient approach. In this context, a novel animated GIF generation method is proposed to reduce the consumption of computational resources and computation time. The use of an audio file instead of an entire video enables us to create

animated GIFs within the limit of acceptable computation time for end-user devices such as an Nvidia Jetson TX2.

The high-level system architecture of the proposed method is illustrated in Fig. 2. It comprises two main parts: the *HLS Server* and the *HLS Client*. The proposed method mainly focuses on the client-side implementation. In the following subsections, the configuration and role of each component of the proposed method are explained.

3.1 HLS server

The first component of the proposed system architecture is the HLS server. The purpose of the HLS server is to smoothly transmit audio files and segments to concurrent users on heterogeneous end-user devices. Internet Information Services (IIS) was selected for this purpose and locally configured on Microsoft Windows 10. IIS supports most network protocols [29]. To reduce potential corruption or loss of packets during transmission [17], all videos are encoded as H.264/AAC Moving Picture Experts Group 2 (MPEG-2) transport stream (.ts) segments using FFmpeg [12]. Each video segment corresponds to approximately ten seconds of playback with a continuous timestamp. Similarly, the list of segments for each video is stored in a text-based playlist file (M3U8) in the playback order of the segments. Along with the segments, the HLS server also contains the audio file (.mp3) of the

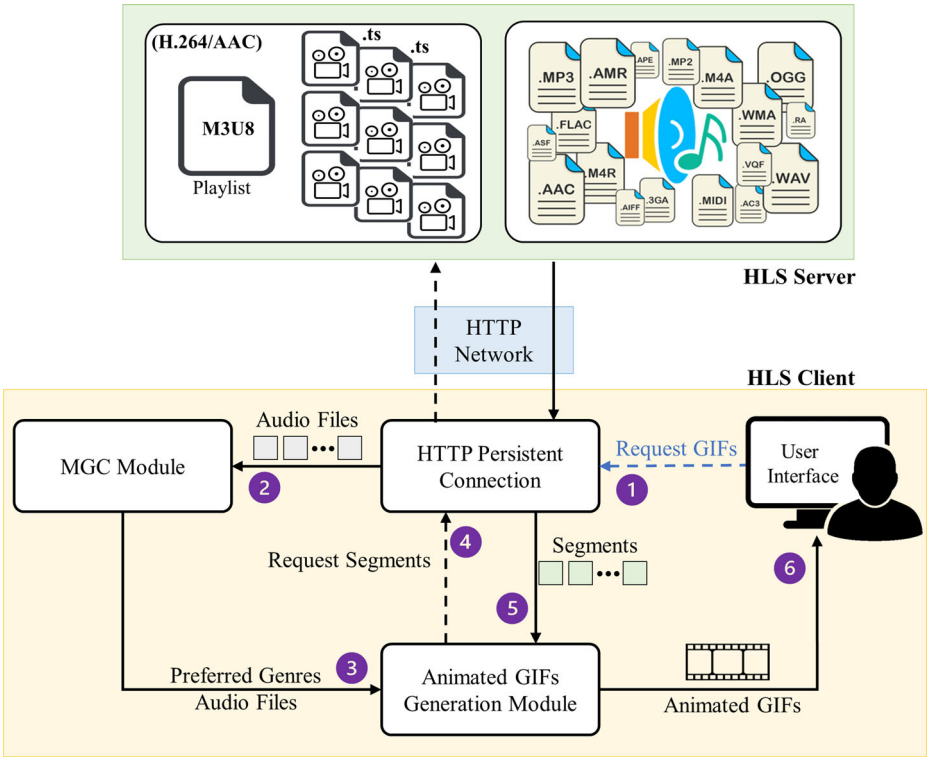

Fig. 2 High-level system architecture of the proposed client-driven GIF generation framework

video, which is separately extracted from the source video using FFmpeg [12]. The detailed hardware specifications of the HLS server are described in Section 4.1.1. The following sections describe the details and roles of each HLS client component.

3.2 HLS client

The purpose of the HLS client is to process the audio file and a segment of the corresponding video to generate a GIF on the end-user device. To this end, the Nvidia Jetson TX2 was configured as the HLS client. This device is a GPU-based board with a 256 core Nvidia pascal architecture [11]. Jetpack 4.3 SDK was used to automate the basic installations and the maximum energy profile used in the proposed method. The HLS client consists of four major components: HTTP persistent connection, MGC module, animated GIFs generation module, and web-based user interface. The details of each component are described in the following subsections.

3.2.1 HTTP persistent connection

Several requests are initiated from the end-user device to obtain music files and segments during the GIF generation process. An HTTP persistent connection is used to download all corresponding files. It is used because it can simultaneously execute multiple requests and returns of data via a single transmission control protocol connection [3]. There are many advantages of using persistent connections, such as fewer new transport layer security handshakes, less overall CPU usage, and fewer round trips [47].

3.2.2 MGC module

The purpose of the MGC module is to analyze the audio files according to user music genre preferences. For this purpose, the GTZAN dataset was used in the experiments, which is extensively used as a benchmark for MGC [38]. The dataset has ten genres, and each genre includes 100 soundtracks of 30 s duration with a sampling rate of 22,050 HZ and a bit depth of 16 bits. The genres are blues, classic, country, disco, hip-hop, jazz, metal, pop, reggae, and rock. The dataset is divided into two sub-datasets: 70% for training and 30% for testing. The Librosa library was used to extract MFCC spectrograms from raw audio data [24]. The extracted features were used as input to the CNN model for training. MFCC spectrograms are a good representation of music signals [7].

The extracted MFCC features were input to the CNN model, which was a two-dimensional array in terms of time and feature value. Each 30 second long audio clip was split into 3 second windows with a size of 19,000 samples $\times$ 129 time $\times$ 128 frequency $\times$ 1 channels. The backbone of the proposed network was based on the VGG16 neural network. The network structure of the music classification model is shown in Fig. 3. The model was trained using the SGDW optimization algorithm with a learning rate of 0.01, momentum rate of 0.9, and the default weight decay value [23]. The training data were fed into the model with a batch size of 256 and learning rate of 0.001 for cost minimization, and 1,000 iterations were performed for learning the sequence patterns in the data. The early stopping method was adopted with ten epochs. The network trained for 100 epochs, and the best validation accuracy was obtained in 51 epochs within 30 minutes training phase. The Keras toolbox was used for feature extraction and to train the

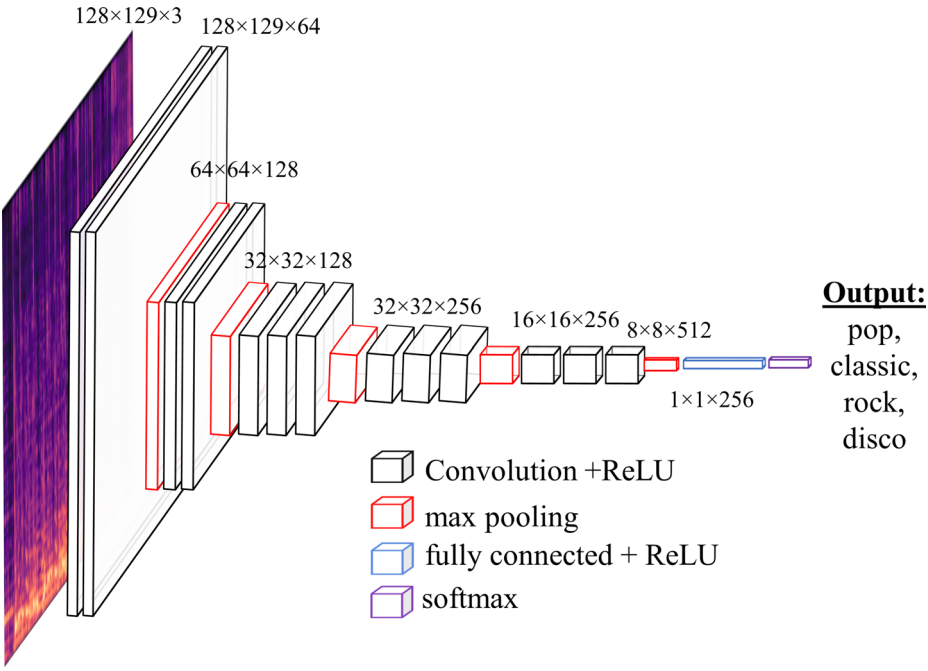

Fig. 3 Network structure of music classification model

GeForce RTX 2080 Ti GPU network. The results of the trained model are described in Section 4.1.

3.2.3 Animated GIFs generation module

The objective of this module is to extract the climax section from the corresponding audio file and estimate the maximum pitch timestamp from it, so that a video segment can be requested to generate the animated GIF. The first three seconds of the segment are used to create a GIF in the proposed technique. Here, the length of each GIF is fixed, but it can be extended to generate GIFs that have a specific length. The proposed method mainly focuses on music videos that have a plot. A composed story generally consists of an exposition, rising action, climax, and resolution. The most exciting part of the plot is the climax section, where all the key events happen, and this represents the most memorable part [16]. Generally, the climax section in a classical story plot begins at 2/3 of the total running time. Figure 4 shows the classical story plot structure of the Big Buck Bunny (2008) video. The details of GIF generation from the climax part using the proposed method are explained in Section 4.2.

3.2.4 Web-based user interface

The user can select the video and preferred music genres and also view the generated GIFs using the web-based user interface. The open source hls.js player is used for this

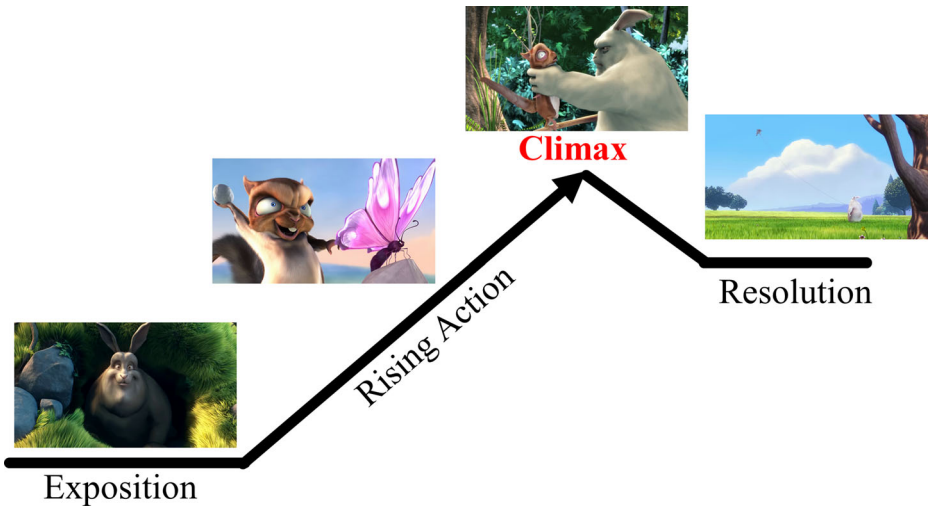

Fig. 4 Classical story plot structure in a video

purpose [10]. Hypertext Markup Language 5 video and media source extensions are needed to play back the transmuted MPEG-2 transport stream. This player supports client-driven data delivery, meaning that the player can decide when to request a segment. The details of the animated GIF generation process are explained in the following sections.

4 Experimental results and discussion

This section presents an extensive experimental evaluation of the proposed method. First, the experimental setup is described along with the baseline approach, which involves processing the entire video and audio. The complete flow of the proposed GIF generation process is then explained from the user perspective. The accuracy of the proposed action MGC model is presented and compared with those of other well-known approaches. Finally, the performance of the proposed method is compared with that of the baseline method.

4.1 Experimental setup

4.1.1 Hardware configuration

In the experimental evaluation, both the HLS clients and the HLS server were locally configured. Two different hardware configurations were used for the HLS client: A high-computational-resources (HCR) device ran on the open-source Ubuntu 18.04 LTS operating system, and a low-computational-resources (LCR) device was configured using an Nvidia Jetson TX2. The proposed and baseline methods were deployed on the HLS clients separately for the LCR and HCR devices. The HLS server was configured on Windows 10 and used in all experiments. All hardware devices were locally connected to the SKKU school network. Table 1 lists the specifications of each hardware device used in the experiments. The entire GIF generation process from the user perspective using the proposed approach is explained in the following subsection.

Table 1 Specification of hardware devices

Device	CPU	GPU	RAM
HLS Server	Intel Core i7-8700K	GeForce GTX 1080	32 GB
HCR Client	Quad-core 2.10 GHz Xeon	GeForce RTX 2080 Ti	62 GB
LCR Client	HMP Dual Denver 2/2MB L2 + Quad ARM A57/2MB L2	Nvidia Pascal 256 CUDA cores	8 GB

4.1.2 Proposed GIF generation process

This section explains the entire flow of the proposed GIF generation process from the user perspective. The flow is explained based on 16 popular videos that were selected from YouTube.[3] A complete description of the videos is provided in Table 2. The statistics for the number of views were collected in June 2020. Because of their popularity, some of the videos have been viewed more than a billion times on YouTube. All the videos used in the experiments had a resolution of 480 × 360 pixels.

The user selects a video using a web-based interface to start the process. The user then selects the music genre preference using a web-based interface. The system requests an audio file for the corresponding video. The downloaded audio file is then analyzed by the proposed trained CNN model according to the user music genre preference. If the music genre of the audio file is consistent with the user preference, the system extracts the climax section from it using the Pydub library [32]. As described in Section 3.2.3, most of the videos follow the same plot structure, and the climax section begins after 2/3 of the running time. Thus, only the last 25% of the audio file is used as the climax section. The model estimates the maximum pitch timestamp from the climax section using Crepe [19]. The timestamp information is obtained in seconds to determine and download a segment. Equation 2 is used to estimate the segment number from the obtained pitch timestamp information.

The system requests a specific segment to be downloaded from the HLS server. Later, that segment is used to generate an animated GIF. The system uses FFmpeg [12] to create the GIF from the segment. Algorithm 1 shows all the processing steps required for using the proposed method to generate a GIF for each video. The variables are Ct (climax time), A (audio file), Sn (segment number), Asr (sample rate of the audio file), Sd (segment duration), and Pt (timestamp of maximum pitch). Sd is a constant and its value is ten units (seconds).

$$Ct = A/Asr \times 0.75 \tag{1}$$

$$Sn = \frac{Ct + Pt}{Sd} \tag{2}$$

[3]The 16 videos were arbitrarily selected for the experimental evaluation. However, the proposed approach is not limited to these videos and can be used for any arbitrary videos.

Table 2 List of videos used for analysis in the proposed method

No	Title	YouTube ID	Duration	Views
1	David Guetta - Titanium	JRfuAukYTKg	4 min 5 sec	1,279,772,497
2	Ed Sheeran - Perfect	2Vv-BfVoq4g	4 min 39 sec	2,436,413,519
3	Ed Sheeran - Shape of You	JGwWNGJdvx8	4 min 23 sec	4,828,112,575
4	Enrique Iglesias - Subeme La Radio	9sg-A-eS6Ig	3 min 52 sec	1,237,101,479
5	Gorillaz - Clint Eastwood	1V_xRb0x9aw	4 min 28 sec	325,611,031
6	Imagine Dragons - Thunder	fKopy74weus	3 min 24 sec	1,465,544,754
7	Luis Fonsi - Despacito	kJQP7kiw5Fk	4 min 41 sec	6,806,190,121
8	Maroon 5 - Sugar	09R8_2nJtjg	5 min 1 sec	3,208,790,733
9	Marshmello Ft. Bastille - Happier	m7Bc3pLyij0	3 min 53 sec	585,963,221
10	Michael Jackson - They Don't Care About Us	QNJL6nfu__Q	4 min 41 sec	703,855,383
11	Michael Jackson - Thriller	rCeZYRzXyHM	7 min 42 sec	3,132,094
12	Pharrell Williams - Happy	ZbZSe6N_BXs	4 min	634,609,305
13	PSY - Daddy	FrG4TEcSuRg	4 min 5 sec	502,203,499
14	PSY - Gangnam Style	9bZkp7q19f0	4 min 12 sec	3,653,550,385
15	Shakira - Waka Waka	pRpeEdMmmQ0	3 min 30 sec	2,514,645,280
16	Tones and I - Dance Monkey	q0hyYWKXF0Q	3 min 56 sec	1,061,327,788

Algorithm 1 Processing steps required for the proposed method.

Input: Video IDs
Output: Animated GIf for the preferred music genre
Initialisation : Number of video selected videos N; User genre preference P
for 1 to N **do**
    Request and download A for video ID
    Identify music genre according to P from A
    if (A is consistent with P) **then**
        Extract climax section from A
        Estimate maximum pitch and its timestamp
        Compute Sn using **Equation 2**
        Request and download the segment
        Generate animated GIf from the segment
    end if
end for

4.1.3 Baseline method

This subsection explains the baseline method used for comparison with the proposed GIF generation method. As described in Section 2, the previous methods used the entire video in the GIF generation process. Here, the entire video and audio are used in the baseline method. As highlighted in Section 4.1.2, the baseline method uses the same web-based interface. In the baseline method, after the video and music genre preferences are selected, the client-side device requests the corresponding video file. The audio file is extracted from the video using FFmpeg [12]; further, the audio file is analyzed using the proposed trained CNN model to

classify the music genre. Further, the baseline model estimates the maximum pitch timestamp from the entire audio file using Crepe [19]. The timestamp information is obtained within seconds to determine the starting point in the video to generate the GIF. Later, the timestamp information is used to generate the GIF from the video using FFmpeg [12]. Algorithm 2 shows the processing steps employed in the baseline method. The computation times of the proposed and baseline approaches are compared in the following experiments.

Algorithm 2 Processing steps required for the baseline method.

Input: Video IDs
Output: Animated GIf for the preferred music genre
Initialisation : Number of video selected videos N; User genre preference P
for 1 to N **do**
    Request and download video for the video ID
    Extract audio from video file
    Identify music genre according to P from A
    if (A is consistent with P) **then**
        Estimate maximum pitch from A and its timestamp
        Generate animated GIf from the video
    end if
end for

4.2 Experimental evaluation of MGC

This subsection evaluates the current CNN methods on the GTZAN dataset. To the best of our knowledge, Senac at al. [36] have achieved the best performance on the GTZAN dataset. The proposed method performed 4.06% better, in terms of the validation accuracy, within 133.83 million floating point operations per second. The experimental results of the baseline method and the proposed method on the GTZAN dataset are shown in Table 3. The proposed CNN model was used in all the experiments to identify the music genre according to user interest.

4.3 Performance analysis of the proposed method

This section compares the performance of the proposed GIF generation method with that of the baseline scheme described in Section 4.1.3. The performance evaluation was conducted

Table 3 Performance comparison on GTZAN dataset

Methods	Acc. %
Schindler, Alexander, et al. [35]	80.30
Zhang, Chiyuan, et al. [42]	82.00
Sigtia, Siddharth, et al. [37]	83.00
Zhang, Pengjing, et al. [44]	83.90
Hamel, Philippe, et al. [15]	84.30
Zhang, Weibin, et al. [45]	87.40
Senac, Christine, et al. [36]	91.00
Proposed	**94.70**

The bold entries show the proposed method performance comparison with other approaches

on 16 videos with different playtimes (see Table 2 for details). The computation time for the baseline method was determined by considering the time required to (i) download the video corresponding to the video, (ii) extract the audio, (iii) identify the genre, (iv) estimate the pitch from the audio, and (v) generate the GIF from the video. Meanwhile, the computation time for the proposed method was determined by considering the time required to (i) download the audio corresponding to the video, (ii) identify the genre, (iii) estimate the pitch from the climax section, (iv) download the segment, and (v) generate the GIF from the segment. Here, the model loading time is not encompassed in all experiments to calculate computation time.

The computation times required (seconds) to generate the animated GIF using the baseline and proposed methods were compared in the first experiment. Both approaches were configured on the HCR device (refer to Table 1 for the detailed specifications of the device). The sizes of the segment and climax section employed to estimate the pitch in the proposed method were significantly smaller than that of the video and the duration of the audio used in the baseline method. The overall computation time of the proposed method was significantly lower than that of the baseline method. Tables 4 and 5 show the computation times required for the HCR device to create a GIF using the baseline method and the proposed method, respectively.

Since this study focused on creating GIFs using the computational resources of the end-user device, in the next experiment, the proposed and baseline approaches were configured on the LCR device (i.e., an Nvidia Jetson TX2). Tables 6 and 7 show the computation times required to create a GIF on the LCR device using the baseline method and the proposed method, respectively. The overall computation times obtained using the proposed method were significantly lower than those obtained using the baseline method.

The combined duration of the 16 videos was 70 min. To generate the 16 corresponding GIFs on the HCR device, the baseline method required 11.27 min, whereas the proposed

Table 4 Computation time needed to create the GIF using the baseline method on HCR device

No.	Download video	Extract audio	Identify genre	Estimate pitch	Generate GIF	Total (seconds)
1	0.69	0.58	16.09	32.43	1.43	51.22
2	0.4	0.62	15.08	33.52	1.15	50.77
3	0.35	0.62	11.57	26.44	1.37	40.35
4	0.57	0.58	10.3	23.27	2.03	36.75
5	0.74	0.61	11.79	27.18	1.48	41.8
6	0.3	0.55	8.85	20.72	0.85	31.27
7	1	0.63	12.25	27.84	1.53	43.25
8	0.7	0.69	13.42	30.68	1.87	47.36
9	0.38	0.5	10.27	23.54	1.16	35.85
10	0.7	0.62	12.27	28.05	2.13	43.77
11	0.62	0.86	19.66	46.22	1.26	68.62
12	0.35	0.58	10.63	23.91	1.37	36.84
13	0.55	0.6	10.81	24.22	1.71	37.89
14	0.92	0.58	11.03	25.22	1.93	39.68
15	0.98	0.52	9.6	21.03	1.55	33.68
16	0.7	0.46	10.54	23.86	1.53	37.09

Table 5 Computation time needed to create the GIF using the proposed method on HCR device

No.	Download audio	Identify genre	Estimate pitch	Download segment	Generate GIF	Total (seconds)
1	0.22	15.01	5.45	0.05	1.54	**22.27**
2	0.24	16.61	4.06	0.05	1.46	**22.42**
3	0.24	16.14	3.87	0.06	1.49	**21.8**
4	0.16	13.66	3.55	0.08	2.63	**20.08**
5	0.22	16.65	4.17	0.06	1.88	**22.98**
6	0.26	13.18	3.25	0.06	0.95	**17.7**
7	0.36	18.25	3.91	0.06	2.18	**24.76**
8	0.25	18.67	4.15	0.08	2.37	**25.52**
9	0.25	14.25	3.4	0.05	1.61	**19.56**
10	0.21	16.71	3.94	0.05	2.18	**23.09**
11	0.4	27.78	6.77	0.05	1.32	**36.32**
12	0.16	14.9	3.67	0.06	1.22	**20.01**
13	0.21	15	3.44	0.066	2.34	**21.056**
14	0.23	15.04	3.64	0.08	3.36	**22.35**
15	0.25	13.29	3.18	0.06	2.85	**19.63**
16	0.21	14.9	3.52	0.06	2.36	**21.05**

The bold entries show the proposed method performance comparison with other approaches

Table 6 Computation time needed to create the GIF using the baseline method on LCR device

No.	Download video	Extract audio	Identify genre	Estimate pitch	Generate GIF	Total (seconds)
1	5.82	1.38	42.75	236.78	2.44	289.17
2	8.8	1.54	36.26	260.27	1.78	308.65
3	8.66	1.23	34.16	244.08	2.36	290.49
4	2.38	1.05	30.7	217.04	3.21	254.38
5	1.98	1.25	35.18	250.24	2.74	291.39
6	4.92	0.94	28.54	188.62	1.17	224.19
7	4.5	1.31	36.36	260.78	2.61	305.56
8	3.76	1.31	39.25	277.7	3.25	325.27
9	3.26	1.03	30.29	214.66	1.85	251.09
10	6.51	1.27	36.31	259.23	3.62	306.94
11	12.49	1.85	59.27	426.42	1.93	501.96
12	2.56	1.1	31.31	222.81	2.15	259.93
13	12.26	1.12	31.54	226.12	3.08	274.12
14	9.86	1.07	32.55	233.26	3.25	279.99
15	12.94	0.97	27.34	195.88	2.56	239.69
16	6.32	0.98	30.46	217.38	2.51	257.65

Table 7 Computation time needed to create the GIF using the proposed method on LCR device

No.	Download audio	Identify genre	Estimate pitch	Download segment	Generate GIF	Total (seconds)
1	2.18	47.07	37.23	0.31	2.52	**89.31**
2	3.29	41.53	33.76	0.14	2.26	**80.98**
3	3	38.8	31.58	0.12	2.57	**76.07**
4	0.43	34.42	28.46	0.67	3.87	**67.85**
5	3.19	39.45	32.51	0.32	2.86	**78.33**
6	1.44	29.99	25.07	0.16	1.38	**58.04**
7	1.62	41.7	33.19	0.28	3.26	**80.05**
8	1.04	43.6	35.49	0.32	3.94	**84.39**
9	2.27	34.03	28.08	0.1	2.51	**66.99**
10	2.29	41.41	33.55	0.11	4.47	**81.83**
11	3.46	66.24	55.13	0.13	2.05	**127.01**
12	0.51	35.42	29.08	0.12	2	**67.13**
13	0.54	35.71	29.48	0.57	3.72	**70.02**
14	3.49	37.34	29.85	1.02	4.99	**76.69**
15	2.64	31.34	25.72	0.31	4.15	**64.16**
16	2.17	34.44	28.71	0.13	3.86	**69.31**

The bold entries show the proposed method performance comparison with other approaches

method required 6.01 min. Furthermore, on the LRC device, the baseline method required 77.67 min, and the proposed method required 20.64 min. Thus, based on the analysis of these 16 videos, on average, the proposed method was 1.87 times and 3.76 times faster than the baseline method on the HCR and LCR devices, respectively. In conclusion, these results show that the proposed method is more computationally efficient than the baseline method on both HCR and LCR devices.

4.4 Qualitative evaluation

This section presents the evaluation of the quality of the GIFs generated using the proposed method by comparing these with those obtained from YouTube and those generated using the baseline approach. The evaluation was based on a survey conducted with the help of 16 participants. Undergraduate students were recruited from our university as participants for this task. They were divided into four groups based on their music genre of interest. Each group of participants was shown three GIFs (i.e. YouTube, baseline, and proposed).

The survey was based on 16 videos (refer to Table 2). The quality of the generated GIFs were evaluated using the exact rating scale. The participants were asked to rate the GIFs according to the arousal aspect. An anonymized questionnaire was created for the generated GIFs so that the users could not determine which method was used (i.e., YouTube or baseline or proposed). They were asked to watch all GIFs and rank them on a scale of 1–10 (1 being the worst and 10 being the best). Table 8 shows the ratings given by the participants for all three approaches. The average ratings obtained for all 16 videos for the YouTube GIFs, the baseline method, and the proposed method were 6.6, 7, and 7.9, respectively. Figure 5 shows the sample frames obtained using the GIFs generated using each of the three approaches.

Table 8 Average rating (1∼10) by the participants

User Group No.	YouTube	Baseline	Proposed
Group 1	6	7	**8.5**
	5.5	6.5	**6.5**
	7.5	7	**8.5**
	6.5	7	**8**
Group 2	7	7	**7.5**
	5.5	6.5	**8**
	5	6.5	**6.5**
	7	7	**8**
Group 3	8	6.5	**8.5**
	7.5	7	**8**
	6.5	6	**7.5**
	7	8	**8**
Group 4	6.5	8	**9**
	6.5	7.5	**8**
	6.5	7	**7.5**
	8	7.5	**8.5**

The bold entries show the proposed method performance comparison with other approaches

4.5 Discussion

The previous sections evaluated the overall effectiveness of the present study by comparing the proposed and baseline approaches. The proposed approach exhibited better performance and reduced computation time on the HCR and LCR devices (Nvidia Jetson TX2). Instead of processing the entire audio and video (baseline), the proposed method used the climax section of the audio and video segment to generate an animated GIF. This is also indicated in the experimental results of comparison of the proposed method with the baseline, according to which the proposed approach was 3.76 and 1.87 times more computationally efficient on the LCR and HCR devices, respectively. This reduces the overall demand for computational resources and the computation time required to generate GIFs on end-user devices.

In qualitative evaluation in Section 4.4, the proposed method received overall a higher average rating than the other approaches used in the qualitative analysis. One of the main

Fig. 5 Illustrations of the frame samples from generated GIFs

reasons that the GIFs generated using the proposed method have higher ratings was because the GIFs were generated from the most exciting parts of the videos.

This study demonstrates the use of an acoustic feature in the GIF generation process while using client device computational resources. Instead of processing the entire audio and video, the proposed method uses a small portion of the audio (climax section) and a video segment to generate the animated GIF. This makes it computationally efficient. There is one constraint while analyzing an acoustic feature. It is possible that while analyzing the baseline and proposed methods to obtain the maximum pitch timestamp, the timestamp information may be the same.

The proposed framework is designed to support a wide range of end-user devices with diverse computational resources capabilities. Because of its simplicity and scalability for implementing various configuration devices [21], the proposed approach can be easily adapted to other animated images (WebP), recommendation techniques [25, 43], and streaming protocols such as dynamic adaptive streaming over HTTP (DASH). In addition to reducing the required server computational resources, the proposed method can serve as a privacy-preserving solution using efficient encryption techniques [28, 30] that can be integrated into other client-driven solutions [26, 27]; moreover, it can be adapted in three-screen TV solutions [33, 34].

5 Conclusion

This paper proposes a novel, lightweight method for generating animated GIFs using end-user-device computational resources for the entire process. The proposed method analyzes the climax section of the audio file, estimates the maximum pitch, and obtains the corresponding video segment to generate the animated GIF. This improves computational efficiency and decreases the demand for communication and storage resources on resource-constrained devices. The extensive experimental results obtained based on a set of 16 videos show that the proposed approach is 3.76 times more computationally efficient compared with the baseline on an Nvidia Jetson TX2. Moreover, it is 1.87 times more computationally efficient than the baseline on the HCR device. Qualitative results show that the proposed method outperforms other methods and receives higher overall ratings.

Acknowledgements This work was supported by a National Research Foundation of Korea (NRF) grant funded by the Korean government (MSIT) No.2019R1A2C1010476. This work was also supported in part by the Gachon University research fund of 2019 (Grant No. GCU-2019-0776).

References

1. Andén J, Mallat S (2014) Deep scattering spectrum. IEEE Trans Signal Process 62(16):4114–4128
2. Bakhshi S, Shamma DA, Kennedy L, Song Y, De Juan P, Kaye J (2016) Fast, cheap, and good: Why animated gifs engage us. In: Proceedings of the 2016 chi conference on human factors in computing systems, pp 575–586
3. Belshe M, Peon R, Thomson M (2015) Hypertext transfer protocol version 2 (http/2)
4. Chen W, Rudovic OO, Picard RW (2017) Gifgif+: Collecting emotional animated gifs with clustered multi-task learning. In: 2017 seventh international conference on affective computing and intelligent interaction (ACII), IEEE, pp 510–517
5. Chiliguano P, Fazekas G (2016) Hybrid music recommender using content-based and social information. In: 2016 IEEE international conference on acoustics, speech and signal processing, ICASSP, IEEE, pp 2618–2622
6. Choi K, Fazekas G, Sandler M (2016) Automatic tagging using deep convolutional neural networks. arXiv:160600298
7. Choi K, Fazekas G, Sandler M, Cho K (2017) Convolutional recurrent neural networks for music classification. In: 2017 IEEE international conference on acoustics, speech and signal processing, ICASSP, IEEE, pp 2392–2396
8. Costa YM, Oliveira L, Koerich AL, Gouyon F, Martins J (2012) Music genre classification using lbp textural features. Signal Process 92(11):2723–2737
9. Dai J, Liang S, Xue W, Ni C, Liu W (2016) Long short-term memory recurrent neural network based segment features for music genre classification. In: 2016 10Th international symposium on chinese spoken language processing, ISCSLP, IEEE, pp 1–5
10. Dev V (2020) Javascript hls client using media source extension. https://github.com/video-dev/hls.js/
11. Developer N (2018) Jetson tx2 module. https://developer.nvidia.com/
12. FFmpeg (2020) Ffmpeg github page. https://github.com/FFmpeg/FFmpeg
13. Gygli M, Soleymani M (2016) Analyzing and predicting gif interestingness. In: Proceedings of the 24th ACM international conference on multimedia, pp 122–126
14. Gygli M, Song Y, Cao L (2016) Video2gif: Automatic generation of animated gifs from video. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 1001–1009
15. Hamel P, Eck D (2010) Learning features from music audio with deep belief networks. In: ISMIR, Utrecht, The Netherlands, vol 10, pp 339–344
16. Hauge M (2017) The five key turning points of all successful movie scripts. http://www.movieoutline.com/
17. Hopkins R (1994) Digital terrestrial hdtv for north america: the grand alliance hdtv system. IEEE Trans Consum Electron 40(3):185–198
18. Jou B, Bhattacharya S, Chang SF (2014) Predicting viewer perceived emotions in animated gifs. In: Proceedings of the 22nd ACM international conference on multimedia, pp 213–216
19. Kim JW, Salamon J, Li P, Bello JP (2018) Crepe: A convolutional representation for pitch estimation. In: 2018 IEEE international conference on acoustics, speech and signal processing, ICASSP, IEEE, pp 161–165
20. Li TL, Chan AB, Chun AH (2010) Automatic musical pattern feature extraction using convolutional neural network. Genre 10:1x1
21. Li Y, Xu G, Ge J, Liu P, Fu X (2020) Energy-efficient resource allocation for application including dependent tasks in mobile edge computing. KSII Transactions on Internet & Information Systems 14(6)
22. Liu T, Wan J, Dai X, Liu F, You Q, Luo J (2020) Sentiment recognition for short annotated gifs using visual-textual fusion. IEEE Trans Multimedia 22(4):1098–1110
23. Loshchilov I, Hutter F (2018) Fixing weight decay regularization in adam. arXiv:171105101
24. McFee B, Raffel C, Liang D, Ellis DP, McVicar M, Battenberg E, Nieto O (2015) librosa: Audio and music signal analysis in python. In: Proceedings of the 14th python in science conference, vol 8
25. Mu R, Zeng X (2020) Auxiliary stacked denoising autoencoder based collaborative filtering recommendation. KSII Transactions on Internet & Information Systems 14(6)
26. Mujtaba G, Ryu ES (2020) Client-driven personalized trailer framework using thumbnail containers. IEEE Access 8:60417–60427
27. Mujtaba G, Kim S, Park E, Kim S, Ryu J, Ryu ES (2019) Client-driven animated keyframe generation system using music analysis. In: Proceedings of the Korean society of broadcast engineers conference, The Korean Institute of Broadcast and Media Engineers, pp 173–175
28. Mujtaba G, Tahir M, Soomro MH (2019) Energy efficient data encryption techniques in smartphones. Wireless Personal Communications 106(4):2023–2035

29. O Leary M (2019) Iis iis iis and modsecurity. In: Cyber Operations, Springer, pp 789–819
30. Qiu S, Xu G, Ahmad H, Xu G, Qiu X, Xu H (2019) An improved lightweight two-factor authentication and key agreement protocol with dynamic identity based on elliptic curve cryptography. TIIS 13(2):978–1002
31. Redwood Shores C (2019) Giphy selects oracle data cloud to measure viewability across billions of gifs. https://www.oracle.com/
32. Robert J (2020) Pydub library. https://github.com/jiaaro/
33. Ryu ES, Jayant N (2011) Home gateway for three-screen tv using h. 264 svc and raptor fec. IEEE Trans Consum Electron 57(4):1652–1660
34. Ryu ES, Yoo C (2008) Towards building large scale live media streaming framework for a u-city. Multimedia Tools and Applications 37(3):319–338
35. Schindler A, Lidy T, Rauber A (2016) Comparing shallow versus deep neural network architectures for automatic music genre classification. In: FMT, pp 17–21
36. Senac C, Pellegrini T, Mouret F, Pinquier J (2017) Music feature maps with convolutional neural networks for music genre classification. In: Proceedings of the 15th international workshop on content-based multimedia indexing, pp 1–5
37. Sigtia S, Dixon S 2014 Improved music feature learning with deep neural networks. In: 2014 IEEE International conference on acoustics, speech and signal processing, ICASSP, IEEE, pp 6959–6963
38. Tzanetakis G, Cook P (2002) Musical genre classification of audio signals. IEEE Trans Speech Audio Process 10(5):293–302
39. Van den Oord A, Dieleman S, Schrauwen B (2013) Deep content-based music recommendation. In: Advances in neural information processing systems, pp 2643–2651
40. Vernallis C et al (2004) Experiencing music video: Aesthetics and cultural context. Columbia University Press
41. Wu MJ, Jang JSR (2015) Combining acoustic and multilevel visual features for music genre classification. ACM Transactions on Multimedia Computing Communications, and Applications (TOMM) 12(1):1–17
42. Zhang C, Evangelopoulos G, Voinea S, Rosasco L, Poggio T (2014) A deep representation for invariance and music classification. In: 2014 IEEE International conference on acoustics, speech and signal processing, ICASSP, IEEE, pp 6984–6988
43. Zhang C, Wu D, Ao L, Wang M, Cai Y (2020) Social-aware collaborative caching based on user preferences for d2d content sharing. KSII Transactions on Internet & Information Systems 14(3)
44. Zhang P, Zheng X, Zhang W, Li S, Qian S, He W, Zhang S, Wang Z (2015) A deep neural network for modeling music. In: Proceedings of the 5th ACM on international conference on multimedia retrieval, pp 379–386
45. Zhang W, Lei W, Xu X, Xing X (2016) Improved music genre classification with convolutional neural networks. In: Interspeech, pp 3304–3308
46. Zhou Y, Song Y, Berg TL (2018) Image2gif: Generating cinemagraphs using recurrent deep q-networks. In: 2018 IEEE Winter Conference on Applications of Computer Vision (WACV), IEEE, pp 170–178
47. Zurawski R (2004) The hypertext transfer protocol and uniform resource identifier. In: The industrial information technology handbook, CRC Press, pp 456–478