Journal of KIIT. Vol. 19, No. 11, pp. 143-153, Nov. 30, 2021. pISSN 1598-8619, eISSN 2093-7571 
http://dx.doi.org/10.14801/jkiit.2021.19.11.143

# Development of a Web Browser-based Character in Video Metadata Generation Tool

Minjeong Kim*, Younsoon Shin**

This research was supported by the MSIT(Ministry of Science, ICT), Korea, under the High-Potential Individuals Global Training Program)(2021-0-01549) supervised by the IITP(Institute for Information & Communications Technology Planning & Evaluation)

## Abstract

Until recently, activating in video platforms and streaming services, the need for managing metadata of videos is increasing day by day to offer deep learning services using these videos. The broadcaster manages the metadata of videos using the 'Digital Asset Management System(DAMS)', a program that saves and manages the metadata of videos. However, since non-professionals cannot use the program for videos they upload directly, they must manually create metadata for each video. But this method is inefficient. Therefore, this paper proposes detecting faces in a video, then recognizing the faces and automatically generating metadata that indexes the person and summarizes the time that the person appeared in the video through a web browser.

## 요 약

최근 영상 관련 플랫폼이나 스트리밍 서비스가 활성화되면서 이러한 영상들을 활용한 딥러닝 서비스를 제공하기 위해서 영상들의 메타데이터를 관리하는 것에 대한 필요성 또한 나날이 증가하고 있다. 방송사에서는 영상 콘텐츠들의 메타데이터를 저장하고 관리하는 프로그램인 'DAMS(Digital Asset Management System)'를 사용하여 영상의 메타데이터를 관리하고 있다. 하지만 해당 프로그램은 일반인들이 직접 업로드하는 영상에서는 사용할 수 없기 때문에, 해당 영상들은 수동으로 직접 각각의 영상에 대한 메타데이터를 만들어야 한다. 그러나 이러한 방법은 비효율적이다. 따라서 본 논문에서는 비디오에서 얼굴을 감지한 후, 인식하여 등장인물을 색인하고, 해당 인물이 영상 내에서 등장한 시간을 정리한 메타데이터를 웹 브라우저를 통해 자동으로 생성하는 방법을 제안한다.

Keywords

metadata, face detection, face recognition, deep learning, indexing

* Dongguk University, Department of Computer Science & Engineering
- ORCID: https://orcid.org/0000-0001-7796-3938
** Dongguk University, Department of Computer Science & Engineering Professor
- ORCID: https://orcid.org/0000-0001-8589-9403

· Received: Oct. 19, 2021, Revised: Nov. 04, 2021, Accepted: Nov. 07, 2021
· Corresponding Author: Younsoon Shin
Dongguk University, Department of Computer Science & Engineering
Tel.: +82-2-2290-1676, Email: ysshin@dongguk.edu

## Ⅰ. Introduction

TV programs use numerous video footage suitable for the situation. These footage are used from as short as a few days ago to as long as decades ago. Broadcasting stations use the Digital Asset Management System(DAMS), a meta-information system that manages video data of the station, to easily search and utilize past video footage. DAMS generates and manages meta-information related to the video, such as the cast, the summary plot, and the description of each scene[1].

Recently, as video platforms such as 'YouTube' and 'Instagram' are activated in addition to TV programs, video data are rapidly increasing. The need for a service that generates metadata of personal video data to provide various deep learning services using these video data is also increasing day by day[2]. However, videos uploaded themselves still need to be manually produced information on the person index. This method is inefficient in terms of time and cost for use these days when image data are accumulated in large quantities[3].

Therefore, this paper proposes a method for easily generating metadata using image data uploaded by ordinary people. By using deep learning to detect and classify the face of the cast, it automatically generates indexed output images for the cast and time data when the cast appeared in the video.

In Chapter 2, we explain the related research and theoretical background. Chapter 3 explains the research method proposed in this paper. The experimental results are shown in Chapter 4, and conclusions and future works are given in Chapter 5.

## Ⅱ. Background and related works

### 2.1 The definition of metadata

'Meta' in metadata means 'higher', and metadata means 'data' in which additional information is added after analyzing and classifying existing information. This definition of metadata is defined differently for each researcher as shown in Table 1[4]. Levy defined it as integrated management data for all forms of digitized information resources generated in a computer environment, and Dempsey&Heerly defined it as an intellectual element that can help users effectively use resources. Domestic researchers also have a similar definition to overseas researchers. In particular, the Korea Communications Commission defined metadata as attribute information and described it as data are given to content according to specific rules to efficiently find and use the information to be found among large amounts of information. In addition, broadcasting contents consist of additional information and essence, including all data (planning, organizing, production, operation, transmission, etc.) used in production and services related to additional information.

The meaning of metadata encompassing these definitions is briefly referred to as 'Data about Data(herry)', and metadata in a broadcast program can be said to be data with various additional information such as the title, genre, and cast information. Metadata mentioned in this paper uses the definition of Metadata in a broadcast program.

Table 1. Definition of metadata by researcher

| Researcher | Definition |
|---|---|
| Levy, 1994 | Integrated management data for all forms of digitized information resources generated in a computer environment |
| Herry, 1996 | Data about data |
| Dempsey & Heerly, 1997 | Intelligent factors that can help users use resources effectively |
| Korea Communicat-ions Commission, 2011 | Including all data (organizing, production, etc.) used for production and services related to additional information on broadcasting content |

### 2.2 Existing metadata generation tools and libraries

Programs can store meta-information on objects currently processed by receiving videos or images as input.

'VATIC[5]' is an online video annotation tool for computer vision research that crowdsources work to Amazon's Mechanical Turk.

'DarkLabel[6]' is a utility program that can label object bounding boxes with ID and name in videos and images. It also can be used to crop videos, sample training images in a video, and mosaic image region. Anyone can use it for noncommercial purposes.

'VoTT[7]' is an open-source annotation and labeling tool for image and video assets. VoTT is a React + Redux Web application written in TypeScript.

'ELAN[8]' is an annotation tool for audio and video recordings. With ELAN a user can add an unlimited number of textual annotations to video recordings. An annotation can be a sentence, word or gloss, a comment, translation, or a description of any feature observed in the media. An annotation can be time-aligned to the media, or it can refer to other existing annotations. The content of annotations consists of Unicode text, and annotation documents are stored in an XML format.

### 2.3 Existing face recognition research

With the recent development of computer vision using deep learning, the Object Detection Algorithm Model is also developing. Table 2 is the result of comparing object detection models with the public SAR-Ship dataset[9][10].

As shown in Table 2, the existing YOLO-v3 was a model with a high Frame Per Seconds(FPS), while the mean Average Precision(mAP) was relatively low. However, YOLO-v5 performs well in both FPS and mAP.

Table 2. Comparison result for the SAR-Ship dataset

| Algorithm | FPS | $AP^{50}$ |
|---|---|---|
| Mask-RCNN | 8.85 | 86.56 |
| YOLO-v3 | 19.23 | 86.58 |
| YOLO-v4-tiny | 76.92 | 79.83 |
| YOLO-v4 | 22.22 | 87.36 |
| EfficientDet-d0 | 13.16 | 86.37 |
| YOLO-v5 | 25.64 | 88.16 |

Therefore, we use YOLO-v5 for learning in this paper because the YOLO-v5 model with the best FPS and Average Precision(AP) among the object detection models is most suitable for real-time processing videos[11].

## III. System design and implementation

This service is a web browser that tags the name of the characters in the video and outputs the timeline of the characters. There are web services and app services in the service method that is easy for users to access, but since large-capacity video files must be processed, it was produced by selecting a web browser method rather than the app method.

### 3.1 The structure of the entire system structure

Fig. 1 shows the configuration of systems used in this service to create a meta-information management system through a web service. The system can be divided into a front-end that receives video from users on a web browser and outputs results and a back-end that processes videos and outputs results.

First, when the user inputs the video through the web browser, the video process part is executed in the back-end. In the video process part, the faces of the character are recognized and indexed in the input video. Then, based on the results, box the person's face on the input video, print the name at the top so that it can be seen at a glance, and store the person's appearance time log separately.

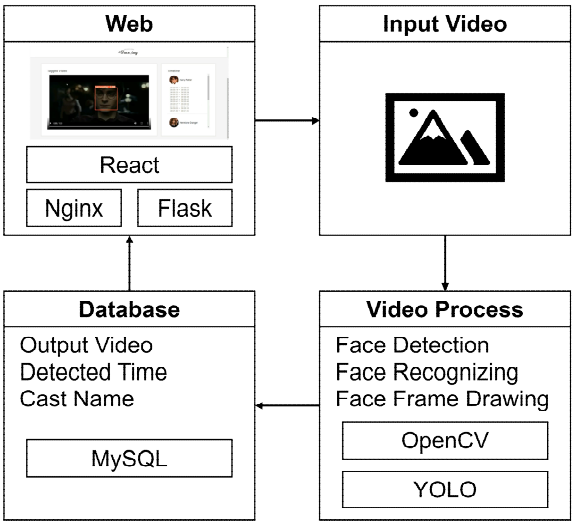


Fig. 1. System architecture

In the data process part, the appearance time log and tag-processed video that occurred as output in the previous process are stored in the database, and the result is then output on the web. At this time, we use Nginx and Flask to output the result processed by the back-end.

## 3.2 The process of collecting and preprocessing learning data

High-quality learning data are required to provide deep learning-based services. In this paper, learning data were generated using 'Cvat' and 'Roboflow'. In order to generate data for training characters, a face of a character was detected by labeling the face of the person in the video.

We collected the data that extracted images in units of 20 frames from the video crawled the cast after looking up cast members on Google. Since the information is similar to the adjacent frames, the possibility of over-fitting of the learning result increases, so data were extracted every 20 frames.

Afterward, as shown in Fig. 2, a total of 4,737 face data were generated by labeling the face in the image using the 'Cvat' site. The image data consists of 1,093 images for Daniel Radcliffe, 1,669 for Emma Watson, and 1,975 for Rupert Grint.

For image learning after labeling, feature points, which are labeled coordinate values, are extracted from the labeled image as a JSON file in the form of YOLO-v5 PyTorch. Of the total 4,737 pictures labeled, 70% of data was used for Train, 20% for Valid, and 10% for Test. As shown in Fig. 3, clustering was performed using the 'Roboflow' site. After labeling and clustering are completed, label.txt containing information of face location in the frame is shown in Fig. 4.

## 3.3 Face recognition process based deep learning

The final goal of this service is to output a video with the name Tag of each cast member attached to the original video and output the time the cast appeared. In order to provide such a service, it is necessary to detect a person's face in the video and then go through a process of recognizing the faces of the cast among the detected faces.

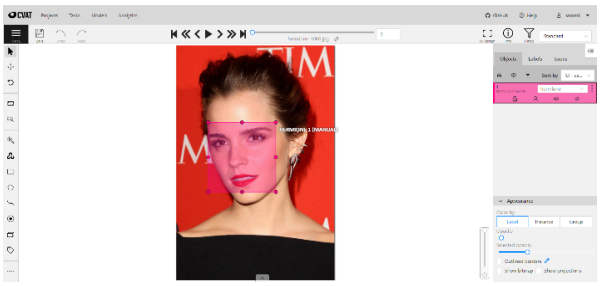

Fig. 2. Example of labeling using 'cvat' site

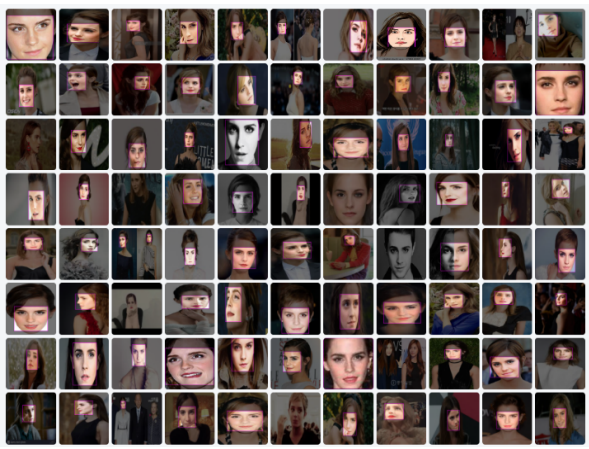

Fig. 3. Labeling result example

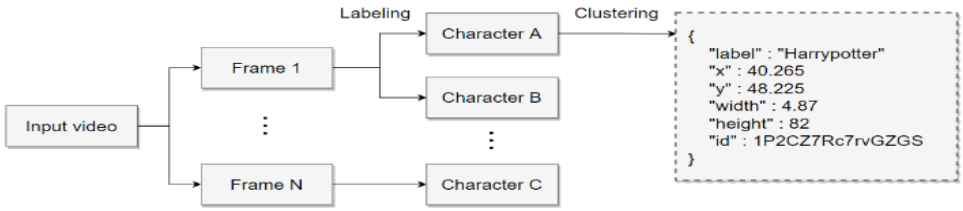


Fig. 4. Process of generating face data for characters

In order to recognize characters in the video using deep learning-based face recognition algorithms and store metadata based on the recognized results, the accuracy of the algorithm must be high. If the accuracy is low, there is little difference between the time to check the data extraction results and the time to collect data manually. Therefore, YOLO, whose performance was verified as a deep learning neural network for object detection, was used.

The YOLO-v5 model offers a total of four models: s(small), m(medium), l(large), and x(xlarge) as shown in Table 3. YOLO-v5-s has the lowest performance, but the processing speed is the fastest because the FPS is the highest. On the other hand, YOLO-v5-x has the highest performance and high accuracy but has the lowest FPS, which is slow. Reducing FPS slows detection, which has the disadvantage of lowering user service satisfaction[12].

We decided to proceed learning with YOLO-v5-x or YOLOv5-l model, emphasizing the performance of distinguishing people rather than speed since learning can be conducted using GPU and is not intended for real-time videos but registered videos.

In addition to model size, image size, batch size, and epoch size are also crucial for YOLO learning[13]. ‘Epoch’ refers to the number of times all datasets have learned about the entire neural network.

Table 3. Comparison of YOLOv5 pre-trained checkpoints

| Model | FPS | Speed | $AP^{test}$ |
|---|---|---|---|
| YOLO-v5-s | 455 | 2.2ms | 36.7 |
| YOLO-v5-m | 345 | 2.9ms | 44.5 |
| YOLO-v5-l | 264 | 3.8ms | 48.2 |
| YOLO-5-x | 167 | 6.0ms | 50.4 |

If the epoch is too small, the possibility of underfitting increases, and if the epoch is too large, the chance of overfitting is increased[14]. The number of batches to achieve 1 Epoch is ‘Iteration’, and the size of data entering a batch is ‘Batch’ size. Since the optimized size was different for each dataset, the test was conducted by changing the value.

The numbers were ‘Image size-batch size-Epoch size-model size’,

(1) 214-16-500-x,
(2) 214-32-300-x,
(3) 214-32-500-x,
(4) 314-16-300-x,
(5) 314-16-500-l,
(6) 416-16-300-x,
(7) 416-16-400-l

and the experiment was conducted with a total of 7 cases. In Fig. 5, the algorithm’s performance was compared using the ‘mAP’ graph. ‘mAP’ is a value mainly used for performance evaluation of a Convolutional Neural Network(CNN) model, and performance can be evaluated by considering both the detection rate and accuracy of the algorithm. The higher the AP, the better the algorithm’s performance. Fig. 6 shows the 90% smoothing of graph figures using exponential smoothing to make it easier to compare performance.

As a result of comparing the graphs, the performance was good in the order of (5), (7), (6), (4), (1), (2), (3). Finally, the model was learned by ‘314-16-500-l’(image size:314, batch size:16, Epoch size:500, and pre-weight model size:l).

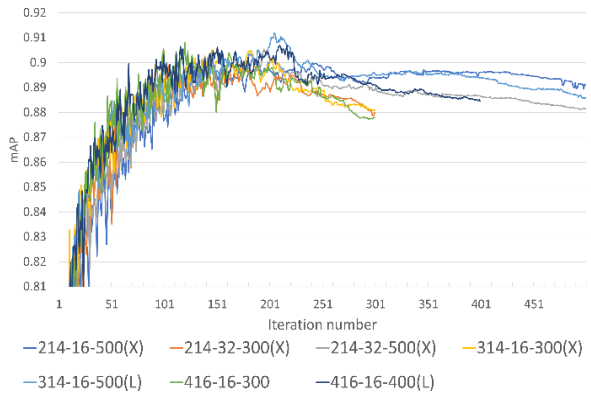


Fig. 5. Comparison of performance using mAP

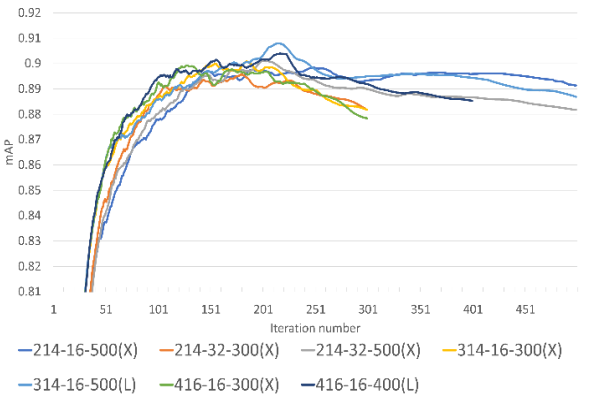


Fig. 6. Comparison of performance using mAP value smoothing 90%


```
{
    "label" : "harry"
    "x" : 40.265
    "y" : 48.225
    "width" : 4.87
    "height" : 82
    "id" : "1P2CZ7R37rvGZGS"
}
```


Fig. 7. Example of the json file format of the labeled dataset

The facial recognition neural network was learned with a COCO dataset labeled with classes of three characters in the previous process. Fig. 7 is an example of the JSON file format returned by 'Rroboflow'. The label represents the class information corresponding to the image, id represents the id of the image, and the bounding box consists of x, y, width, and height.

After detecting the character's faces using the YOLO model, the function of outputting the time characters appear was implemented. (1) is an equation for calculating the time when a character appears in the video. The equation for calculating the total playtime of the video is 'full FRAME number/total video FPS'. At the moment of recognizing the face based on the equation, the FPS of the video was divided into the number of the frame number to obtain the time(seconds) within the video.

$$\frac{detected\ \ FRAME}{input\ \ video\ \ FPS} \tag{1}$$

The frame at the moment of detecting the character's face can be obtained to calculate the time (second). However, since the video has multiple frames consecutively, face tracking is implemented in a way that continues to appear when a person of the same class appears in a continuous frame by storing the frame number immediately before. After tracking, the time at the earliest point of time was saved as start and the last time as an end.

The start and end times were stored in a list in the character's class, and when the video processing was completed, the list was handed over to Flask to print out the timeline in which the character appeared.

Fig. 8 is a diagram showing a face recognition algorithm. First, when the video inputs, the total FPS of the video is calculated, and the character's face detection is started using the deep learning model learned with the previously collected data. The box is processed using a 'draw frame' on the input video when the character's face is detected. Thereafter, the progress time of the frame is checked using 'get_frame' and stored together with the name of the detected person in the log. The detect process is repeated if the current number of frames is less than the total number of frames based on the total number of frames.

Tthe database as shown in Table 4 that stores the log information largely consists of 'id', 'cid' storing a character class as a number, 'start', the time when the character first appeared, and 'end', the last time the character appeared in the corresponding scene.

Table 4. Comparison of YOLOv5 pre-trained checkpoints

| Field | Type | NULL | Key |
|---|---|---|---|
| id | varchar | NO | PRI |
| cid | int | NO | |
| start | varchar | NO | |
| end | varchar | NO | |

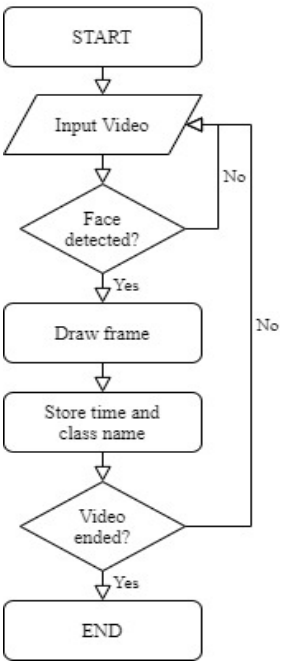


Fig. 8. Face recognition system

After checking all frames and analyzing the video, store the recorded log in MySQL database in the form of [‘id(auto increment)’, ‘class number’, ‘start time’, ‘end time’] and store the output video in Amazon S3 Bucket.

## IV. Experiment

In order to measure the accuracy of the method proposed in this paper, we experimented with some scenes from a movie involving three actors whose faces were previously learned.

### 4.1 Environment of experiment

Table 5 and 6 are evaluation environments for measuring the performance of distinguishing casts in a video.

Table 5. Hardware system environment used for performance evaluation

| Type | Specification |
|---|---|
| CPU | Intel CORE i5 $10^{th}$ Gen |
| MEMORY | DDR4 16GB |
| OS | Windows 10 |

Table 6. Software system environment used for performance evaluation

| Type | Version |
|---|---|
| YOLO | v5 |
| Python | v3.7 |
| OpenCV | v4.5.3 |
| learning (training/valid/test) data | 4,737 images of 3 characters’ face (Daniel Radcliffe, Emma Watson, Rupert Grint) |
| test video | 2min 30sec video (Youtube - Harry Potter and the Deathly Hallows Pt.1&2 \| Official Trailer) |

The test video data used for performance evaluation are a total of four videos within 60 seconds and a scene from a movie featuring all three actors, Daniel Radcliffe (cast: Harry), Emma Watson (cast: Hermione), and Rupert Grant (cast: Ron), was prepared.

### 4.2 Experiment

The meta-information management system for characters proposed in this paper uses facial data of previously registered characters to provide metadata on which characters in the input video appeared at what time through the website. First, access the website in a PC or mobile environment and upload the Input video as shown in Fig. 9.

Fig. 10 shows a page after uploading the video and pressing the ‘Tagging Cast’ grey button to execute the image process. When the process is completed, the tagged video on the left is the output video with the name tagging on the cast’s face on the input video. In addition, in the timeline on the right, the start time and end time in which each cast appeared are output. When a timeline marked in the form of ‘hh:mm:ss’ is clicked, the output video moves to the corresponding time.

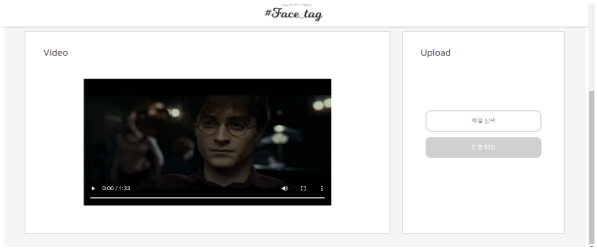


Fig. 9. Complete upload video file

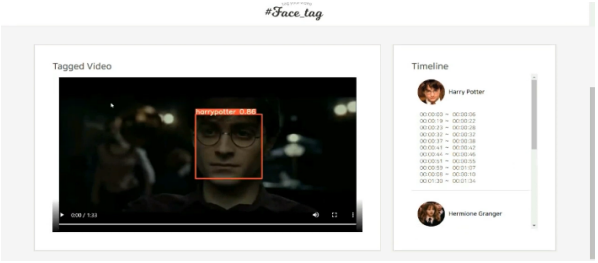


Fig. 10. Complete face detection and print timeline

## 4.3 Performance evaluation

For the performance evaluation of the system, the average value of recognition accuracy that distinguished three characters was compared. Table 7 is a table comparing the average accuracy of characters for each scene. The average accuracy was calculated by adding all the accuracy values of the recognized faces between the start-time and end-time and then dividing the number of frames of the character. Additionally, if the face recognition accuracy is lower than 50%, it is considered that the face recognition is not properly recognized, and the frame is not output.

Table 7. Comparison of the average face recognition accuracy of each scene

| Field | Harry | Hermione | Ron |
|---|---|---|---|
| Forward face-1 (Fig 11) | 0.88 | N/A | N/A |
| Forward face-2 (Fig 12) | N/A | 0.89 | N/A |
| Forward face-3 (Fig 13) | 0.79 | <0.5 | 0.83 |
| Not forward face (Fig 14) | 0.85 | N/A | N/A |
| Close up face (Fig 15) | 0.75 | N/A | N/A |
| Far away face (Fig 16) | 0.76 | N/A | N/A |
| Shadowed face (Fig 17) | <0.5 | <0.5 | <0.5 |
| Focus out face (Fig 18) | <0.5 | 0.87 | <0.5 |
| Over 90° aspect of face-1 (Fig 19) | <0.5 | N/A | N/A |
| Over 90° aspect of face-2 (Fig 20) | 0.59 | N/A | N/A |
| Obscured face (Fig 21) | <0.5 | N/A | N/A |

Fig. 11, 12, 13 is a scene in which a character looks forward. It can be seen that the accuracy is 88%, 89%, and 83%, in the order of characters Harry, Hermione, and Ron.

On the other hand, in Fig. 14, where the face is detected sideways because the person does not look forward, the accuracy of Harry is 85%, which is 3% lower than the accuracy of the front face. Most of the trained datasets were frontal images of the face, but the accuracy of the side of the face rotated below 45 degree is not significantly different. This is because when the face is rotated below 45 degrees, there is no significant difference from the position of the feature point of the front face.

As can be seen by comparing the accuracy of Fig. 15 and 16, the distance of the face does not significantly affect the accuracy.

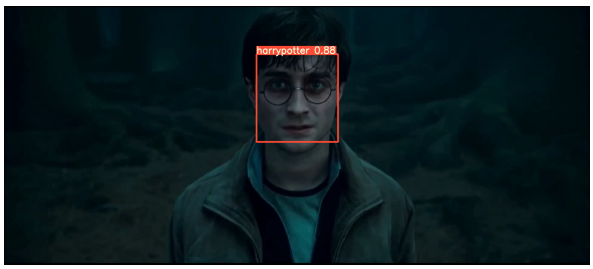


Fig. 11. Forward face - 1

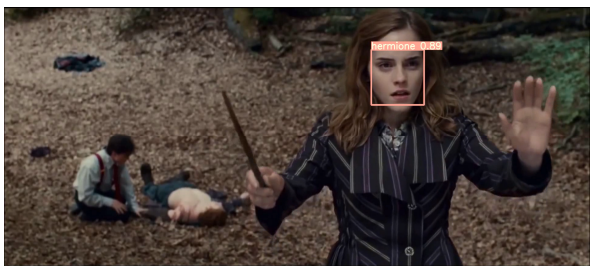


Fig. 12. Forward face - 2

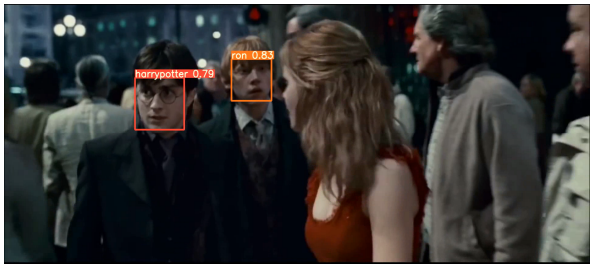


Fig. 13. Forward face - 3

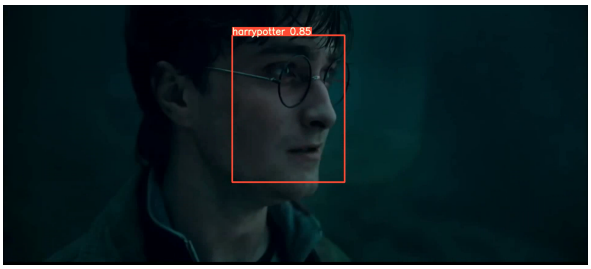


Fig. 14. Not forward face

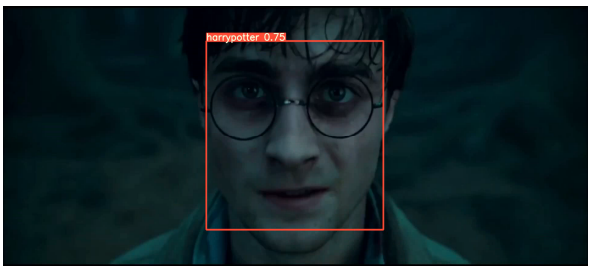


Fig. 15. Close up face


Fig. 16. Far away face

Fig. 17. Shadowed face

Fig. 18. Out of focus face

Fig. 19. Face rotated more than 90 degrees - 1

The face recognition accuracy of the characters in Fig. 17 is less than 50%, and as shown in Fig. 17, it can be seen that the accuracy is significantly reduced if the face is small and there is a shadow. This result suggests that shadows are a factor that has a more significant impact on accuracy than face size because shadows make it more challenging to distinguish facial features.

Fig. 18 is a scene where the focus is off the Ron's face, and the accuracy of Ron is less than 50%. The accuracy of the scene is at least 33% lower than the accuracy of 83% of the Ron's forward face. When out of focus, a phenomenon similar to the blur effect occurs. Face recognition is performed by extracting features based on a combination of fine texture information on the face.

However, the feature points disappear, when the blur effect occurs, and the facial recognition performance decreases. In addition, face of Harry was not out of focus, but moved too fast, so the accuracy of face recognition was low.

Fig. 19 is a scene in which the face rotates more than 90°, so only less than half of the face appears. Fig. 20 is a scene in which about half of the face is shown by looking forward a little more. In Fig. 19, face recognition was measured to be less than 50% as only a part of the side of the face appeared. The reason is that only half of the features held by the front face remain when the side is more than 90 degrees, since most of the learning data sets are front face images. However, if it exceeds half of the face, as shown in Fig. 20, it can be seen that the accuracy increases to 59%.


Fig. 20. Face rotated more than 90 degrees - 2

Fig. 21. Face obscured by an obstacle

Fig. 21 is a scene in which a part of the face is covered by a hand obstacle surrounding the face. The face recognition accuracy is less than 50%. The accuracy is at least 9% lower than the side face accuracy of 59% in Fig. 20. This occurs because obstacles cover the face and eliminate the facial feature values.

## V. Conclusion

In this paper, we propose how to automatically generate metadata of the cast's index and appearance time by recognizing faces between casts based on deep learning.

The entire service was implemented as a website to increase accessibility.

The proposed method uses YOLO-v5 to recognize the pre-trained faces of the casts, check whether the casts appeared in the video and if so, save when they appeared in the video. The accuracy of indexing the faces of characters by detecting and classifying the character's face using a Deep learning system was up to 89%.

Experiments using video data when the same cast appeared in different scenes showed that small changes in face size or small rotation angle(Fig. 14, 21) did not significantly affect accuracy. But shadows(Fig. 17), blurring(Fig. 18), obstacles in front of the face(Fig. 21), and large face rotation angles(Fig. 19) significantly affected the accuracy of the person recognition.

Face recognition models capable of strongly performing even in various situations may be combined and used to minimize this effect. There are various models, such as a model resistant to lighting changes using histogram qualification, a deblurring algorithm that removes blur, an algorithm that can improve resolution, and a model resistant to changes in face angle that use 3D image data[15].

In future research, based on the above algorithm, it plans to modify it to be robust to the characteristics of the video and to add a system configuration that can store meta-information such as the plot in the video, the facial expressions of the cast, and lines.

## References

[1] Adrianne Lee and Hee-Jung Kim, "A study of metadata element design for broadcasting records management-based on the case study of MBC TV program records", Journal of the Korean Library and Information Science, Vol. 43, No. 3, pp. 269-295, Sep. 2009. http://dx.doi.org/10.4275/KSLIS.2009.43.3.269.

[2] Jinseung Kim, Yongkoo Han and Youngkoo Lee, "Efficient storage and retrieval for automatic indexing of persons in videos", Journal of Korea Multimedia Society, Vol. 14, No. 8 pp. 1050-1060, Aug. 2011. http://dx.doi.org/10.9717/kmms.2011.14.8.1050.

[3] Minji Kang, Jaekeun Cho, Giseok Choe, and Jongho Nang, "A semi-automatic authoring tool on HTML5 web browser for generating broadcasting content metadata", The Korean Institute of Information Scientists and Engineers, Vol. 46, No. 1, pp. 1507-1509, Jun. 2019.

[4] Sungho Kwak, "A study on a standardization model of Korean broadcasting metadata system", Department of Communication The Graduate School of Sogang University, Aug. 2012.
[5] VATIC https://github.com/cvondrick/vatic [accessed: Sep. 15, 2016]
[6] DarkLabel https://github.com/darkpgmr/DarkLabel [accessed: Sep. 15, 2021]
[7] VoTT https://github.com/Microsoft/VoTT [accessed: Sep. 15, 2021]
[8] ELAN https://archive.mpi.nl/tla/elan [accessed: Sep. 16, 2021]
[9] Y. Wang, C. Wang, H. Zhang, Y. Dong, and S. Wei, "A SAR dataset of ship detection for deep learning under complex backgrounds", Remote Sensing, Vol. 11, No. 7, pp. 765, Mar. 2019. http://dx.doi.org/10.3390/rs11070765.
[10] G. Zhang, Z. Li, X. Li, C. Yin, and Z. Shi, "A novel salient feature fusion method for ship detection in synthetic aperture radar images", IEEE, Vol. 8, pp. 215904-215914, Nov. 2020. https://doi.org/10.1109/ACCESS.2020.3041372.
[11] G. Yang et al., "Face mask recognition system with YOLOV5 based on image recognition", 2020 IEEE 6th International Conference on Computer and Communications (ICCC), Chengdu, China, pp. 1398-1404, Dec. 2020. https://doi.org/10.1109/ICCC51575.2020.9345042.
[12] YOLOv5 https://github.com/ultralytics/yolov5 [accessed: Aug. 31, 2021]
[13] Thuan Do, "Evolution of YOLO algorithm and YOLOV5: the state-of-the-art object detection algorithm", 2021.
[14] D. Garg, P. Goel, S. Pandya, A. Ganatra, and K. Kotecha, "A deep learning approach for face detection using YOLO", 2018 IEEE Punecon, Pune, India, pp. 1-4, Dec. 2018. https://doi.org/10.1109/PUNECON.2018.8745376.
[15] Hyeong-Ill Ki, Seung-Ho Lee, and Yong-Man Ro, "Trends of facial recognition technology in the wild environment", The Journal of The Korean Institute of Communication Sciences, Vol. 31, No. 4, pp. 88-98, Mar. 2014.

## Authors

Minjeong Kim

Feb, 2019 ~ present : BS degree in Department of Computer Science & Engineering, Dongguk University
Research interests : computer vision, IoT, computer network

Younsoon Shin

1999 : B.S. in Computer Science and Statistics
2011 : Ph.D. in Information Communication Engineering at Dongguk University
2012 ~ present : Professor at Dongguk University
Research interests : wireless sensor networks, embedded system, and IoT