ISSN 2394-3777 (Print)
ISSN 2394-3785 (Online)
Available online at www.ijartet.com

International Journal of Advanced Research Trends in Engineering and Technology (IJARTET)
Vol. 8, Special Issue 1, August 2021

ML-based Solutions for Designing an End-to-End Video Describing Service

Ameya Khale
Dept. of Computer Engineering
St. John College of Engineering and Management
Palghar, Maharashtra, India
ameyak@sjcem.edu.in

Kanishk Singh
Dept. of Computer Engineering
St. John College of Engineering and Management
Palghar, Maharashtra, India
kanishks@sjcem.edu.in

Aditi Raut
Dept. of Computer Engineering
St. John College of Engineering and Management
Palghar, Maharashtra, India
aditir@sjcem.edu.in

Roshan Gupta
Dept. of Computer Engineering
St. John College of Engineering and Management
Palghar, Maharashtra, India
roshan@sjcem.edu.in

***Abstract*—Video content is created, stored, and consumed at an extremely fast pace in the modern world. Databases and warehouses currently store a very large amount of video data. This data is raw and unstructured in nature and thus would greatly benefit from a service that could read, understand, analyze and describe the data. In this paper, we study the services currently being provided for video data as well as the models that are trained for the purpose of video captioning. We also present an architecture and workflow dedicated towards creating a scalable, cost-effective, and efficient video description service. Our proposed workflow uses modern cloud services and pipelines to provide machine-generated video descriptions to allow indexing, enable smart video feeds and improve accessibility of any video that is hosted on the web, a local device or coming in as a stream. In order to increase accessibility, we also incorporate translation and text-to-speech services to provide regional language access and audio descriptions for the visually impaired. This workflow employs the pre-trained Bi-modal Transformer (BMT) [13] neural network model in order to generate video captions and also assess its performance in real-world use-cases.**

***Keywords*—Accessibility, Audio Descriptions, Cloud Computing, Machine Learning, Video Captioning**

I. INTRODUCTION

Image Processing tasks with the use of sophisticated neural networks has been one of the areas of study in Artificial Intelligence and Machine Learning in recent years that has yielded immensely promising results. Multiple applications such as object detection, labelling, action recognition, etc. have been made possible and are currently in widespread use. The ability to generate descriptions of video content is highly valuable for automated tasks such as labelling and tagging videos. Automated tagging and indexing of videos could help improve the results displayed by a video search engine as well as help generate summaries of thousands of hours of video content. Moreover, it could help visually impaired people to have better experience while accessing, interacting and consuming rich content on the web.

Of the numerous applications mentioned above, many have proven useful to the physically challenged or ones with special needs. Currently, YouTube and other streaming platforms use speech-to text algorithms on the audio to identify spoken words as well as sound effects to generate captions in real-time [1]. In an era of powerful processing on the cloud, extensive fiber network connections and tremendous amounts of online content consumption, accessibility is a major factor in determining the reach of a business as it addresses the most basic requirements of a significant segment of the worlds' population.

With advancements in the field of Image Captioning, there has been growing interest in generating video descriptions using deep learning technologies. Large datasets which contain video and their corresponding human generated captions, such as the ActivityNet [11] dataset have made it possible to train captioning and description generating models on large quantities of video data. The most popular example of this is the dense video captioning task [12] which aims to generate temporal captions based on video features. Multiple models have been trained to achieve this, which employ transformation, bi-directional evaluation, and other innovative methods. Bi-modal Transformer is one of the more advanced models trained to perform dense video captioning. The bi-modal transformer takes into account both the video and audio track of the input to generate richer descriptions. It makes use of a proposal generator model to detect events and a caption generator model to generate time-stamped descriptions of the events occurring in the video. We employ this model as the engine of our proposed application and architect a cloud-

based workflow around that which is scalable, efficient and accessible to the general audience.

II. Related Work

We survey existing solutions and how Machine Learning models are deployed in the cloud. In this section. We also provide background information about existing systems/ platforms which provide machine learning inference as a service.

A. Growth of Cloud Computing

Compute power has been getting cheaper and cheaper in the last decade, to the point where it's more convenient to use ad-hoc virtual machines to do compute-intensive tasks than buy, manage, and maintain physical machines. Developers have taken advantage of the cloud to train and optimize models online [5]. They also design data pipelines to continuously train and optimize their models which are served to real-world applications, all from the cloud. This enables low-latency inference at scale. The trained models are served over a REST API enabling a number of different clients like web and mobile apps to consume them.

B. ML-as-a-Service for Video

We compare the offerings by the three most prominent cloud service providers, namely Google Cloud Platform (GCP), Microsoft Azure and Amazon Web Services (AWS) in Table I. Google provides two AI-based Video services, namely AutoML Video Intelligence and Video Intelligence API as a part of their Google Cloud Platform offerings. The AutoML Video Intelligence service provides the user with a graphical interface and has two main tasks: classification and object tracking. It can classify the various parts of a video based on user-defined labels. Object tracking includes detection of objects and tracking them through the video. The Video Intelligence API provides all of these features and more, with extra features such as explicit content detection, optical character recognition, closed captioning, recognizing specific objects such as faces, logos, celebrities, tracking, auto-labelling, OCR, etc. Both of these products support streaming video data and can be used to send in live video feeds for analytics.

Microsoft with their Azure Media Services provides a Video Indexer, which outputs a JSON that includes data such as labels, faces, shots, written text, brands, sentiments, celebrities, etc. They also provide a web-based graphical console using which the user can interact with this data and create a sophisticated media workflow.

AWS's Amazon Rekognition Video service has a similar feature set which includes detection of objects, activities, scenes, text, faces, celebrities, inappropriate content, etc. Rekognition Video is also able to perform these tasks on an incoming stream, similar to Google.

These services are satisfactory for most common use-cases such as live-streaming, content moderation and delivery, analytics, etc. However, they may not be able to satisfy certain specific consumer requirements such as detailed descriptions of the actions and events that occur in the video. In-depth descriptions can be extremely beneficial for making audio-visual content accessible on the internet, video indexing and search based on events, summarizing long length detecting malicious activities on surveillance cameras, etc. Rekognition can be paired with Amazon Transcribe to generate captions and transcripts in real-time for streaming video.

III. Proposed System

In this section, we outline our approach, discuss the model we have selected and how our system, illustrated in Fig. 2 will work once deployed to the cloud.

A. Bi-modal Transformer

Bi-Modal Transformer is a model which can utilize data from both the visual as well as audio tracks in order to generate a list of captions for the video. These captions are temporal and thus suitable for chronologically describing and by extension summarizing the video.

TABLE I. COMPARISON OF FEATURES PROVIDED BY CLOUD PLATFORMS FOR VIDEO INTELLIGENCE

Feature	Cloud Service Provider		
	GCP	Azure	AWS
Graphical Interface or Dashboard	Yes	Yes	Yes
Labelling	Yes	Yes	Yes
Object Detection	Yes	Yes	Yes
Object Tracking	Yes	Yes	Yes
Person Detection	Yes	Yes	Yes
Person Tracking	No	No	Yes
Closed Captions	Yes	Yes	Yes
Native Live Stream Analysis	Yes	No	Yes
Event/Activity Description	No	No	No
Video summarization	No	No	No
Native Multi-lingual support for labels and objects	No	No	No

The model requires audio features as extracted by a VGGish model, video features as extracted using a pre-trained Inflated 3D model and Stanford NLP's pre-trained GloVe model for word corpus.

The ability of the model to accept both audio and video data enables it to create more rich, descriptive captions as it is able to take as input more data than a traditional model that only uses the visual elements.

The use of GloVe enables the model to have a large corpus of natural language words with their corresponding vector representations. This ensures that the caption sentence generated aptly captures the event or activity it describes.

B. Queues

The queue backend used in our system is Redis. Redis is an open source, in-memory data structure store, used as a database, cache, and message broker [6]. python-rq is a simple Python library for queueing jobs and processing

them in the background with workers [7]. It utilizes Redis as a backend.

After the video is received by the server, it is enqueued into the feature queue. The feature extraction workers are continuously polling this queue. As soon as a new video is enqueued, the processes start the feature extraction task. If the feature extraction worker is busy with another video, the moment it becomes free, it polls the queue and starts processing the next video in the queue.

After features have been extracted, the server enqueues a description generation job into the description queue, and provides the paths to the feature files to the job. Like the feature extraction workers, the captions workers also poll the queue and start generating the captions.

Queues help deal with the large number of incoming files and allow asynchronous processing of data. This improves the scalability of the application.

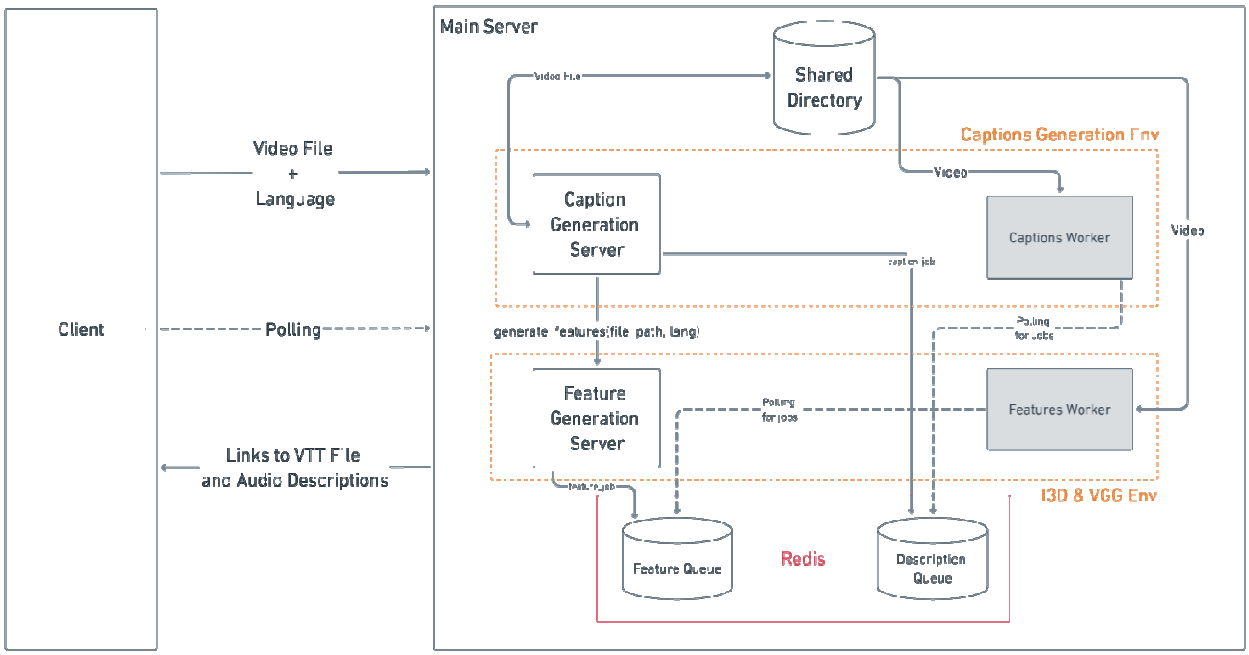

Fig. 1. Cloud-based architecture of the proposed video describing service

C. Backend Server

Our proposed solution employs a combination of 2 server instances to carry out the end-to-end video captioning task. The first server faces the client, and hosts BMT, our trained proposal and caption generator model as well. On initial request, it accepts the video and language choice from the request body and stores the video to the shared directory.

It then passes the video file name and language choice to the second server, the feature generation server, which returns a job ID. This job ID is then returned to the user who uploaded the video.

After features are generated, the captions for the video are generated by the second server, i.e. the caption generating server, and it uses the captions to generate VTT file containing the timestamped captions, and an MP3 file containing captions in a synthesized voice, in a language of the user's choice.

The feature extracting and caption generating servers are built using Flask. Flask is a microframework written in Python which enables the creation of web applications. Flask aims to keep the core simple but extensible, and does not include a database abstraction layer, form validation or any other components for which pre-existing libraries exist [8]. It allows us to build simple as well as complex web applications and leaves the decision of selecting third-party libraries to the developer.

1. Feature Extracting Servers

They are composed of Video and Audio Worker processes. The bi-modal transformer takes video features in the form of I3D features. I3D is a convolutional neural network model for video classification trained on the Kinetics dataset [9]. We use a PyTorch implementation of I3D to generate the I3D features for the video input. The BMT model also requires audio features. These features are generated using VGGish. VGGish is a pre-trained Convolutional Neural Network from Google [10]. The architecture of this network is inspired by the famous VGG networks used for image classification. We use a Tensorflow implementation of VGGish to generate audio features.

The feature extraction server accesses the video from the shared directory. After generating the features, they store them back in the shared directory.

2. Caption Generating Server

The caption generator loads the pre-trained proposal and caption generating models into memory when the system starts.

After feature generation, the caption generation server loads the features generated previously and generates the VTT file and the audio file of the captions in the choice of user's language, which is generated by the Google TTS service. The paths of the VTT file and the audio file are then returned as a JSON response to the client which is polling the server for status of the captioning job.

D. Client

The service is headless in nature, i.e., it is not limited to one client and can communicate with any program acting as the client. The client uploads the video to the Primary Server, which returns a job ID to the client. The client then continuously polls the Primary Server for status of the captioning job. Along with the video, the client also has the option to send a language, as visible in the demo client in Fig. 2.

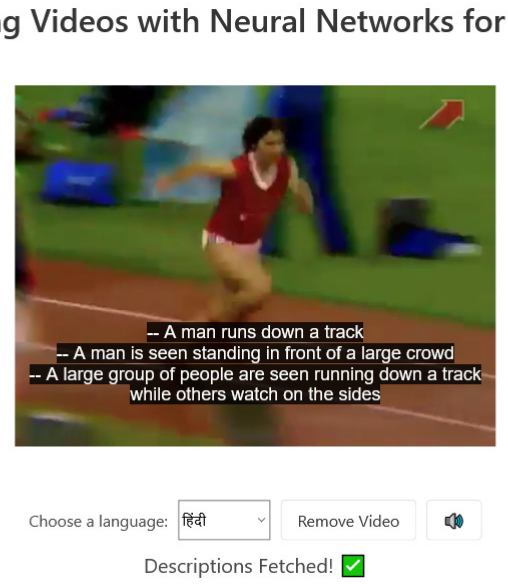

Fig. 2. A user has uploaded a video and selected the Hindi language. The application fetches the VTT captions for it, which are shown to be embedded in the player, and also the audio descriptions in Hindi, which plays when the user presses the speaker icon in the UI.

This language can then be used by the backend to translate the captions to, and also generate text-to-speech for it. The text-to-speech audio can then be played by the client. This allows a potential use of such a system to work as an accessibility mechanism for those who are unable to consume the video content visually.

In case the backend fails to generate captions, the 'status' field of the response reads 'failed', notifying the client that the job has ended in failure, and the client can notify the user through an error message and sound notification.

IV. Results and Future Scope

We have created a React [14] application to demonstrate the working of this system. React is a JavaScript library to create dynamic user interfaces.

```
[
  {
    "start":0.1,
    "end":4.9,
    "sentence":"We see a title screen"
  }
]
```

Fig. 3. Structure of video description for a single event given by the BMT model. The start and end fields denote starting and ending timestamps in seconds.

Using React allowed us to create an application that is minimal and automate tasks such as sending network requests when the user has selected a video and a language.

This reduced the number of steps required from the user's side, making it easier for those using screen readers, etc. to access it and thus more accessible.

The application allows the user to upload a video from their local machine. As soon as the video is selected, it is also uploaded to the backend where it starts getting processed. Once the descriptions are successfully generated the backend transforms the JSON output from the model into the Web Video Text Tracks Format (VTT), and if a language is specified by the user, it also creates audio descriptions for the descriptions in the specified language. Figure 3 shows a sample JSON output of a single event by the BMT model. These are returned to the client, which dynamically adds the VTT descriptions to the video track and enables a button to play the audio descriptions.

VTT is the recommended standard set by the W3C for serving closed captions over the web. VTT is compatible

with a large number of screen readers which are utilized by people with visual impairments.

The implementation can be made more cloud native by using a microservice architecture and taking advantage of managed offerings from cloud providers like AWS. The feature and caption generation servers could be independent microservices, using a blob storage service like AWS S3 to store and share video files and their corresponding feature files. These microservices will listen to their respective queues, which could be AWS SQS instances. Additionally, these microservices can use a database like DynamoDB to store state of the jobs, ensuring that captions generate after features for it have been generated.

V. CONCLUSION

In this paper we conducted a comparative study on the existing machine learning based services offered for video. We discussed the various use-cases of these services and the areas where they could possibly fall short, namely video descriptions and summarization. We discussed the dense video captioning task, the models that can perform it and also employed the BMT model into a proposed cloud-based workflow for a video description generating service. We concluded that the technology is still in its infancy and that we should see many improvements with newer research and models which would improve the overall performance of the system in terms of video description accuracy. There are yet many areas in which the end-to-end system could improve, but overall shows promise with a lot of potential in real-world applications.

E. REFERENCES

[1] Simonite, Tom. “Machine Learning Opens Up New Ways to Help People with Disabilities.” MIT Technology Review, MIT Technology Review, 2 Apr. 2020,

[2] Krishna, Ranjay, et al. “Dense-captioning events in videos.” Proceedings of the IEEE international conference on computer vision. 2017.

[3] Zhou, Luowei, et al. "End-to-end dense video captioning with masked transformer." Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2018

[4] ran, Du, et al. "Learning spatiotemporal features with 3d convolutional networks."Proceedings of the IEEE international conference on computer vision. 2015.

[5] Zhang, Chengliang, et al. "Mark: Exploiting cloud services for cost-effective, slo-aware machine learning inference serving." 2019 {USENIX} Annual Technical Conference ({USENIX}{ATC} 19). 2019.

[6] “*Redis*” 17 Mar. 2021. Accessed on: 6 Apr. 2021 [Online] Available: https://redis.io/

[7] “RQ: Simple job queues for Python” 1 Apr. 2021. Accessed on 6 Apr. 2021. [Online] Available: https://python-rq.org/

[8] “Foreword - Flask Documentation (1.1.x)” 5 Apr. 2020. Accessed on: 6 Apr. 2021. [Online] Available: https://flask.palletsprojects.com/en/1.1.x/foreword/

[9] “I3D model | DeepMind” 12 Feb 2018. Accessed on: 7 Apr. 2021. [Online] Available: https://deepmind.com/research/open-source/i3d-model

[10] Hershey, Shawn, et al. "CNN architectures for large-scale audio classification." 2017 ieee international conference on acoustics, speech and signal processing (icassp). IEEE, 2017.

[11] CabaHeilbron, Fabian and Escorcia, Victor and Ghanem, Bernard and Carlos Niebles, Juan “ActivityNet: a large-scale video benchmark for Human Activity Understanding” Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). June 2015

[12] Krishna, Ranjay and Hata, Kenji and Ren, Frederic and Fei-Fei, Li and Niebles, Juan Carlos “Dense-captioning events in videos” International Conference on Computer Vision (ICCV). 2 May 2017

[13] Iashin, Vladimir, and EsaRahtu. "A better use of audio-visual cues: Dense video captioning with bi-modal transformer." *arXiv preprint arXiv:2005.08271* (2020).

[14] Brian Sam Thomas, Rajat Dogra, Bhaskar Dixit, Aditi Raut. “Automatic Image and Video Colourisation using Deep Learning” 2018 International Conference on Smart City and Emerging Technology(ICSCET), Mumbai, 2018

[15] “React – A JavaScript library for building user interfaces” 1 Apr 2021. Accessed on: 6 Apr. 2021. [Online] Available: https://reactjs.org/