Performance Analysis of Applying Deep Learning for Virtual Background of WebRTC-based Video Conferencing System

Sangwoo Ryu
Graduate School of Artificial Intelligence
POSTECH
Pohang, Korea
rswoo@postech.ac.kr

Kyungchan Ko
Computer Science and Engineering
POSTECH
Pohang, Korea
kkc90@postech.ac.kr

James Won-Ki Hong
Computer Science and Engineering
POSTECH
Pohang, Korea
jwkhong@postech.ac.kr

***Abstract*—With the advancement of artificial intelligence(AI) technology, AI is being used in various industries such as factory automation and autonomous driving. Video conferencing systems have also added functions that use AI to overcome the limitations of existing algorithms, for example, super resolution and virtual background functions using image segmentation. However, web-based video conferencing limits the application of these features due to a limited web browser environment. In this paper, we introduce several approaches to apply deep learning in a web browser environment to provide the features that use deep learning models, and introduce image segmentation models used for virtual background functions in each method and evaluate their performance. Finally, we discuss areas that need to be considered to apply deep learning models to web-based video conferencing.**

***Index Terms*—Artificial Intelligence, Deep learning, Web-based Video Conferencing System, Virtual Background, Image Segmentation, Quality of Experience**

I. Introduction

Due to COVID-19, people have not been able to have face-to-face meetings both in education and businesses. People have been using video conferencing tools such as Zoom [25], Webex [26] and Vmeeting [17] for online work or education since March 2020. To satisfy customers' requirements and increase the satisfaction of using the service, the video conferencing service providers have been adding various additional features. Some of the features show better performance when deep learning [1] is applied than the classical approach [24]. Deep learning can be applied to many features like super resolution to improve the quality of the video, face alignment to show the front side of the face, virtual background to change the background of the users.

Video conferencing systems that require installation on users' devices can directly use programming languages like Python/C++ and easily apply deep learning using a lot of ML libraries. On the other hand, since most of the web applications are provided through JavaScript, deep learning should be applied in a more restrictive environment in web-based video conferencing systems. In this paper, among the many application cases of deep learning, we focus on the virtual background feature. We introduce the methods for applying a deep learning model that performs image segmentation to apply the function to the web environment according to the application locations: server and client. Especially for the client side application, we perform performance evaluation by applying deep learning models to Vmeeting [17], which is a WebRTC [13] based video conference system. Finally, we summarize the areas that need to be considered in applying deep learning models and present the necessary studies to improve future performance and user experience.

II. Background

A. Applying Deep Learning to Web Browser Environment

The way to apply deep learning to the web browser can be divided according to location of the application. When a deep learning model is located in the server, deep learning models can operate using server resources (CPU, GPU) and server-side code independently of the browser, existing methods can be used. However, additional bandwidth usage is required because input and output data must be exchanged with the client side. This can be costly for service providers to build servers and reserve bandwidth. In addition, privacy issues can arise depending on the type of data, and delays in real-time processing may exist due to computations on the server.

When a deep learning model is located in the client, deep learning models can operate using the client device's resources. However, different clients use different devices, they may have different user experiences.

Below we describe two methods of applying deep learning models on the client side.

a) JavaScript Library: Google Open Source Project TensorFlow [2] provides the TensorFlow.js library for machine learning on the web. TensorFlow.js provides pre-trained models for typical use cases like image classification, object detection, and so on. TensorFlow supports multiple backends for storage and math operations, including CPU, WebGL, WASM(Web Assembly) [18]. JavaScript can be extended to mobile environments via React Native [19] and IoT devices via

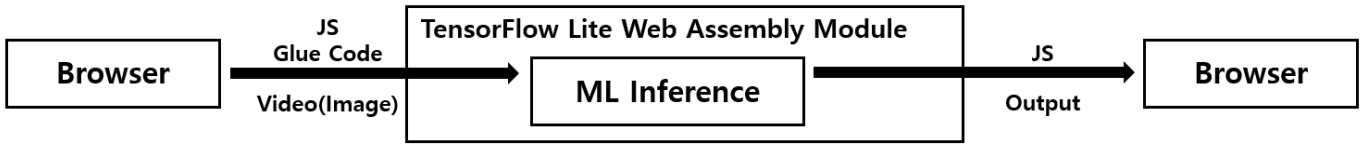

Fig. 1. Example of TensorFlow Lite Web Assembly Pipeline

Node.js [20] as well as browsers. In addition to TensorFlow.js, JavaScript libraries such as Keras.js and ConvNetJS provide tools for applying deep learning to web browsers [3].

b) WebAssembly: Web assembly [4] is a technology that runs code written in C/C++ on the web by converting the code into binary format and executes it on a virtual machine. For example, we can build TensorFlow Lite [21], a machine learning framework for mobile and IoT devices written in C++, into a web assembly module, or build programs containing the entire inference process, including those libraries. Then we load the module from JavaScript and use a model. Web assembly generally shows high speed for computation than pure JavaScript. Google Meet built a Mediapipe, Google Open Source framework for machine learning application to real-time images, as a web assembly, and performed background separation for the virtual background function using a deep learning model [5]. Web assembly is currently supported by browsers such as Chrome, Safari and Firefox since 2017 and are supported by approximately 90% of devices [6]. Because it is used over JavaScript, it can be used on mobile/IoT devices using React Native or Node, as can the JavaScript library.

Fig. 1 shows the pipeline when TensorFlow Lite is built and used as a web assembly. Additional options such as single instruction, multiple data (SIMD) and multi-threads can be used when we use web assembly to accelerate deep learning operations.

B. Image Segmentation

Image segmentation is the task of outputting a pixel-by-pixel mask by judging the location, shape, and pixel of the object in the image. Image segmentation can be used in autonomous driving to distinguish roads, cars, people, etc. or in medical images to determine the location of specific tumors. In video conferences, it can be used for virtual background functions that separate a person from background to create masks and draw backgrounds. Fig. 2 shows an example of a virtual background applied to video conferencing in practice.

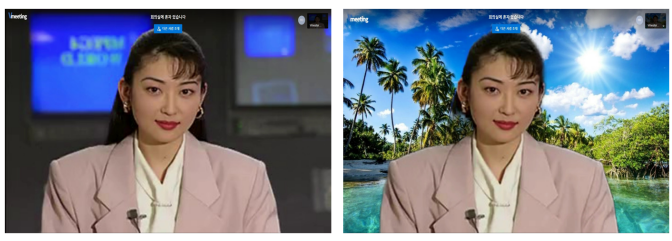

Fig. 2. Left - Original, Right - Virtual Background

DeepLab [7, 8] is a representative deep learning model for image segmentation. MobileNet [9-11] which is specialized in mobile environments is also constantly being studied.

C. Web-based Video Conferencing System

Fig. 3 shows the basic structure of a WebRTC-based video conferencing system. It consists of a signaling server that informs the connection point of clients by exchanging signals using Session Description Protocol (SDP) [12], and a relay server that delivers data such as video and audio.

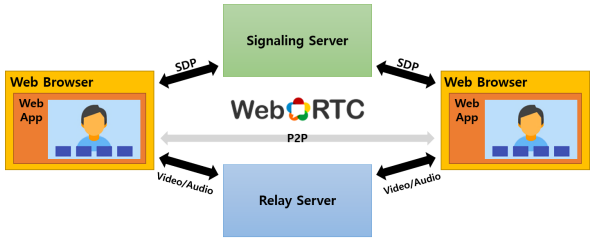

Fig. 3. Architecture of WebRTC-based Video Conferencing System

III. Performance Analysis

This section presents a theoretical bandwidth usage when a deep learning model is applied to a server, and the performance of various application methods when a deep learning model is applied to a Vmeeting client.

A. Server-side Application

When applying a deep learning model to a server, performance is determined by the performance of the server's resources (CPU, GPU) because the client's resources are not used. However, since data needs to be transferred to the server and transferred back to the participating client, tasks directly related to resolution, such as super resolution (SR) [22, 23] directly affect the bandwidth usage on the server. Prior calculation of bandwidth usage is required to perform such tasks. For tasks such as background separation for virtual background functions, the bandwidth is not affected because it is not related to the resolution.

B. Client-side Application

When applying a deep learning model to a client performance is determined by the performance of the client device's resources. As with server applications, depending on the task and the method, it affects bandwidth usage. In case of super resolution (SR), when performing SR on the sender side, both bandwidths are affected from the client to the server and from the server to the client. On the other hand, there is no

TABLE I
PERFORMANCE OF TENSORFLOW.JS MODELS WEBGL BACKEND

MobileNet-V1								
Multiplier	**QuantByte=2**				**QuantByte=1**			
	Size	***FPS***	***Infer.Time (ms)***	***IoU***	***Size***	***FPS***	***Infer.Time (ms)***	***IoU***
1	˜6MB	8.15	80.47	0.9510	˜3MB	8.95	79.01	0.3012
0.75	˜2MB	9.01	69.45	0.9518	˜1MB	8.54	70.63	0.8760
0.5	˜1MB	9.59	61.4	0.9074	˜0.6MB	10.52	57.53	0.7961
ResNet50								
-	**QuantByte=2**				**QuantByte=1**			
	Size	***FPS***	***Infer.Time (ms)***	***IoU***	***Size***	***FPS***	***Infer.Time (ms)***	***IoU***
	˜45MB	3.96	202.36	0.9648	˜22MB	3.98	205.27	0.9649

TABLE II
PERFORMANCE OF MODEL LOADED USING WEBASSEMBLY

Google Meet Segmentation Model								
Model Input Size	**non-SIMD**				**SIMD**			
	Size	***FPS***	***Infer.Time (ms)***	***IoU***	***Size***	***FPS***	***Infer.Time (ms)***	***IoU***
256x144	400KB	14.59	19.08	0.9786	400KB	18.1	6.96	0.9786
190x96	400KB	17.52	9.39	0.9761	400KB	19.07	3.4	0.9761
DeepLabV3								
-	**non-SIMD**				**SIMD**			
	Size	***FPS***	***Infer.Time (ms)***	***IoU***	***Size***	***FPS***	***Infer.Time (ms)***	***IoU***
	2.7MB	3.12	256.9	0.9148	2.7MB	8.34	66.69	0.9148

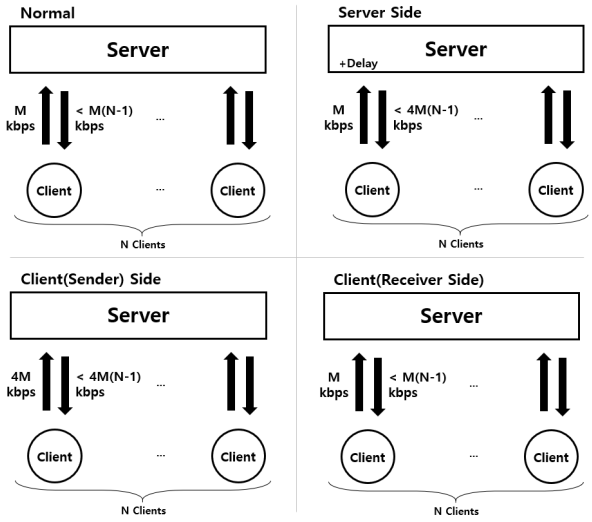

Fig. 4. Expected Bandwidth Usage in 2x Super-Resolution Operation

significant difference from general video conferences when performing SR only on the video of the person displayed on the screen on the receiver side.

Fig. 4 shows expected bandwidth usage of conducting 2x SR, which doubles the width and height of images with N participants and the default bandwidth transmitted to the video conference server at M kbps for the 4 cases: Normal case, applying on the server side, on the client(Sender) side, and on the client(Receiver) side.

Below we show the performance of applying the image segmentation model, which can be provided or used in each application method, to Vmeeting which is a video conference based on WebRTC [13]. This model does not significantly affect bandwidth because it does not change the resolution of the image.

FPS (Frame per Second) and model inference time used an average of 100 seconds of results performed on Chrome browsers on a 2.90 GHz i5-9400F CPU, 16.0 GB RAM desktop PC. Model size used information from official documents. We used a virtual camera provided by the live streaming support program OBS Studio [14] and repeatedly played a 10-second-long "Akiyo" video sequence, which is similar to a video conference in which only the upper body comes out in an indoor environment. IoU(Intersection over Union) is used as accuracy metric of image segmentation task and calculated using MATLAB `jacarrd` function. It used an average of 5 frames of video sequence(1s, 3s, 5s, 7s, 9s).

a) JavaScript Library - TensorFlow.js: TensorFlow.js provides Body-pix [15] as a model for person-background separation, ResNet50 [16] and MobileNet-V1 [9] can be selected as deep learning models and some parameters can be adjusted manually to resize the model. In addition to the models provided, custom models can be loaded and used if necessary.

There are many factors that determine the speed and accuracy of deep learning models. For example, quantization is a method of quantifying float parameters with fewer bits to speed up computation and memory access, and it can affect model size, inference speed, power consumption, and accuracy. Also, the depth of the convolution operation affects the number of parameters and thus the accuracy. Table I shows the performance of the background separation task in video conferencing when applying MobileNet-V1 and ResNet50, both using a WebGL backend. QuantBytes represents the number of bytes for weight quantization, and Multiplier is a parameter that determines the depth of convolution operations. The table shows smaller quantification and the larger the depth of the convolution operation, the larger the model size.

With MobileNet-V1, a larger model tends to have slow inference time but higher accuracy (IoU) and with ResNet50, there was no meaningful difference with the change in quant-Byte.

b) WebAssembly: To use Web Assembly, a custom model is needed that works with the machine learning library to use. For example, TensorFlow Lite provides pre-trained models for typical machine learning tasks that can work with TensorFlow Lite.

Table II shows the performance of the background separation task in video conferencing when applying improved MobileNetV3-small model that is provided by Google Meet under the Apache 2.0 License and DeepLabV3 [7] that is provided by TensorFlow Lite.

The Google Meet Segmentation model has shown better performance than the relatively large DeepLab-V3 model because according to the Google Meet Segmentation model documentation, it uses channel attention method to make the model friendly to CPU inference, uses quantization, thus the total number of parameters is 193K. Both models show better speed with same accuracy when using SIMD.

Among different models, while DeepLabV3 and Google Meet Segmentation model uses encoder-decoder structure, MobileNetV1 and ResNet uses fully convolutional networks, which the upsample process is not trainable and may result in poor accuracy.

Vmeeting is currently using Google Meet Segmentation model for virtual background feature.

IV. Conclusion & Future Work

In this paper, for the application of deep learning in a limited web environment, we described the application methods of deep learning models in web-based video conferencing according to two application locations: servers and clients. In particular, for the client side application, we introduce two approaches, JavaScript libraries and Web assemblies and analyzed bandwidth usage and performance with each approach.

Various factors must be considered when applying deep learning to web-based video conferencing. First, since resource and network bandwidth usage may vary depending on the task, it is necessary to analyze them in advance and select the appropriate model application location. In addition, since the machine learning inference methods/libraries introduced in this paper vary in speed depending on model size and computation volume, the model application method should be chosen for the model we intend to use. The deep learning model research to optimize the model is also being done, not just to boost performance. Since image processing must be performed in real time in web services such as video conferencing, deep learning model selection is required considering both performance and optimization. Finally, it is necessary to study methods to accurately and naturally show the outputs of deep learning models.

Since the performance of the model depends on client device performance when using deep learning models on the client side, we need to study the application of deep learning models that can adapt to the environment to avoid harming the user experience in real-time video conferencing environments. At the same time, models for low-performance devices, such as MobileNet models, also require continuous improvement research.

Acknowledgment

This work was supported by the Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korean government (MSIT) (2018-0-00749, Development of Virtual Network Management Technology based on Artificial Intelligence) and the ITRC (Information Technology Research Center) support program (IITP-2021-2017-0-01633).

References

[1] LeCun, Y. et al. "Deep learning," Nature 521, pp. 436–444, 2015, (https://doi.org/10.1038/nature14539).
[2] Martín Abadi et al., "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015, (https://tensorflow.org).
[3] Ma, Y. et al.. "Moving Deep Learning into Web Browser: How Far Can We Go?," World Wide Web Conference, pp. 1234–1244, Association for Computing Machinery, 2015.
[4] Haas, A., et al. "Bringing the Web up to Speed with WebAssembly," SIGPLAN Not., 52(6), pp. 185–200, 2017.
[5] Background features in google meet, powered by web ml. (2020, October 30). Retrieved March 23, 2021, from https://ai.googleblog.com/2020/10/background-features-in-google-meet.html
[6] Can I use... support tables for HTML5, css3, etc. (n.d.). Retrieved March 23, 2021, from https://caniuse.com/wasm
[7] Chen, L.C. et al. "Rethinking Atrous Convolution for Semantic Image Segmentation," arXiv e-prints, arXiv:1706.05587, 2017.
[8] Chen, L.C. et al. "Encoder-Decoder with Atrous Separable Convolution for Semantic Image Segmentation," arXiv e-prints, arXiv:1802.02611, 2018.
[9] Howard, A. et al. "MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications," arXiv e-prints, arXiv:1704.04861, 2017.
[10] Sandler, M. et al. "MobileNetV2: Inverted Residuals and Linear Bottlenecks," arXiv e-prints, arXiv:1801.04381, 2018.
[11] Howard, A. et al. "Searching for MobileNetV3," arXiv e-prints, arXiv:1905.02244, 2019.
[12] Handley, Mark et al. "SDP: Session Description Protocol," IETF, doi:10.17487/RFC4566, RFC 4566, July 2006.
[13] A. Bergkvist et al. "WebRTC 1.0: Real-time communication between browsers," W3C Working Draft, Feb. 2015.
[14] OBS studio, https://obsproject.com/
[15] Tensorflow. BodyPix - Person Segmentation in the Browser, https://github.com/tensorflow/tfjs-models/tree/master/body-pix
[16] He, Kaiming et al. "Deep Residual Learning for Image Recognition," pp. 770-778. 10.1109/CVPR.2016.90, 2016.
[17] Vmeeting, https://vmeeting.io
[18] Haas, A. et al. "Bringing the Web up to Speed with WebAssembly," SIGPLAN Not., 52(6), pp. 185–200, 2017.
[19] React Native, https://reactnative.dev/
[20] Node.js, https://nodejs.org/
[21] TensorFlow Lite, https://www.tensorflow.org/lite
[22] Yang, W. et al.. "Deep Learning for Single Image Super-Resolution: A Brief Review," arXiv e-prints, arXiv:1808.03344, 2018.
[23] Anwar, S. et al. "A Deep Journey into Super-resolution: A survey," arXiv e-prints, arXiv:1904.07523, 2019.
[24] Alom, Md. Zahangir et al. "A State-of-the-Art Survey on Deep Learning Theory and Architectures," Electronics. 8. 292. 10.3390/electronics8030292, 2019.
[25] Zoom, https://zoom.us/
[26] Webex, https://www.webex.com/