BuzzStream: A real-time video streaming and ML-based image processing system

Gabriel A. Lopez Lugo, Huan Xu

Table of Content

1. Motivation

The COVID-19 pandemic sparked an unprecedented surge in online interactions, creating a demand for sophisticated and interactive online meeting software, which served as an *application pull*. As a response, technology titans like Zoom, Microsoft Teams, and Google Meet introduced real-time video streaming software, marking an impressive leap in *technology push*. Nevertheless, real-time video streaming is an ongoing challenge, with a new wave of application pull emerging to incorporate AI image processing models into online meeting platforms. For instance, Zoom has implemented AI background removal features, while TikTok utilizes image style transfer models. However, it is still an open problem when it comes to blending these functionalities seamlessly, particularly when dealing with system deployment on edge devices, which have limited computational and memory resources. In this context, resource contention becomes a considerable hurdle as both codec and ML inference significantly utilize GPU. Currently, there is no comprehensive open-source solution available for these challenges.

In this project, we aim to explore the design space of video streaming and ML-based image processing systems. We plan to develop "BuzzStream," a video chat web application, employing innovative technologies such as WebRTC and Tensorflow.js. Our goal is not only to address the issues mentioned above but also to achieve a sub-second latency for our video streaming with AI effects.

2. Related Work

The existing work surrounding the blend of online video streaming and real-time ML-based video processing is plentiful and diverse. It spreads across closed-source solutions, open-source streaming solutions, ML-based video processing alternatives, and research efforts aimed at bridging the gap.

2.1 Closed-sourced Solutions

Understanding the functionality of industry leaders provides us a benchmark for crafting a strong, competitive solution. Zoom, as one of the leading platforms for video conferencing, has shown a streamlined approach to real-time video interaction. Another impactful closed-source solution comes from TikTok, a platform that has revolutionized short video content creation and sharing. However, the technical solutions offered by Zoom and TikTok are proprietary, with access to their services gated behind an account registration process.

2.2 Open-source Streaming Solutions

Besides commercial platforms, several open-source solutions support online video streaming. These solutions offer transparency, performance, and the potential for customization, which is particularly significant in our context. For instance, WebRTC emerged as a powerful tool for

facilitating browser-based communication, including audio, video, and data transfers. However, while WebRTC is indeed an impressive new standard, it must be noted that in order to meet the evolving demands of present-day applications, it is essential to integrate it with real-time ML serving infrastructure.

2.3 Open-source ML-based Video Processing Solutions

In addition to streaming, machine learning has introduced transformative changes in the field of video processing. Notably, many open-source projects focus on AI-driven video processing. Although the machine learning community frequently uses PyTorch to train and exhibit their models, it's important to note that PyTorch is not browser-compatible. Additionally, there is an observable shift in the industry towards the use of increasingly larger models in the pursuit of accuracy, an approach which is not conducive to real-time inference tasks. Despite these challenges, we plan to use tools like TensorFlow.js to convert PyTorch-based models to run in a browser setting. This approach allows us to explore and experiment with the optimal balance between accuracy and latency. By achieving this delicate balance, our aim is to ensure real-time functionality without compromising too much on accuracy.

2.4 Bridging the gap

Notable advancements have been made in the separate domains of online video streaming and real-time machine learning-based video processing. Nonetheless, the distinctive value of our study lies in synergizing these two fields and constructing a functional liaison between them. The objective is to integrate solutions such as WebRTC with sophisticated tools like TensorFlow.js, which enable PyTorch-based machine learning models to operate in a browser context. Moreover, we aim to strike a balance between model accuracy and execution speed to assure real-time functionality. This comprehensive application streamlines online video streaming and deploys ML-based video processing capabilities on the fly. The potential applications for such a tool are extensive, spanning from telecommunications and online education to the multimedia industry. Thus, this innovative fusion heralds a captivating new direction in the realms of video streaming and machine learning.

3. Proposed Work

BuzzStream is a dynamic web-based application that provides a platform for users to create, participate in, and enjoy engaging video streaming-based interaction. The comprehensive suite of features accessible through BuzzStream enhances user experience within digital communication spaces, catering to a diverse range of needs. Here are some key features that BuzzStream provides:

3.1 Fully Interactive Video Chat Rooms

Within BuzzStream, users can conveniently create or join virtual platforms known as video chat rooms. These arenas foster live interaction, where individuals can conduct meetings, hangouts,

workshops, or any other forms of group communication. Effectively, video chat rooms within BuzzStream are redefining digital realm dynamics by achieving real-world interaction experiences in a virtual space.

3.2 Smooth Screen Sharing

BuzzStream encompasses a seamless screen sharing feature, boosting the value of digital communication. Be it presentations, tutorials, or cooperative work, this feature ensures everyone is on the same page, promoting mutual understanding and collective decision making. A simple interactive interface facilitates the ease of executing a screen share.

3.3 AI-Based Video Stream Enhancement Tools

The application stands out for its advanced AI-powered effects that can be incorporated into video streams, contributing to a high-quality, engaging video experience. These features allow users to manipulate their backgrounds, effectively removing any unwanted or distracting elements. In addition, facial alteration effects in real-time boosts user confidence about their on-camera appearance, ultimately promoting active participation.

4. Sample Scenarios

In this section, we will explore three different scenarios showcasing the capabilities of BuzzStream in facilitating cross-device interaction and utilizing advanced AI features.

4.1 Cross-Device Interaction Scenario

Consider John, a project manager who is working from home on his personal laptop, while his colleague Sarah, a software analyst, is in the office using her desktop. They have to urgently discuss the workflow of a software release process.

In this scenario, using BuzzStream's in-browser application, John and Sarah can seamlessly join a video call. John, working from his laptop, pops up right from his current browser. In parallel, Sarah, utilizing her office desktop, can easily join the same video chat without needing to switch devices or install any additional software.

During the discussion, Sarah needs to walk John through a complex workflow diagram. She readily shares her screen using BuzzStream's intuitive screen share feature. Despite being on different devices, John views this diagram in real time, making it significantly easier to understand and provide his inputs. The seamless in-browser application compatibility of BuzzStream enables this efficient cross-device collaboration.

4.2 Advanced AI Feature Utilization

Consider an educational scenario where Kayla, a college professor, is conducting a virtual class using BuzzStream. Amid the class, she needs to explain a complex theory using diagrams.

Through the comprehensive AI-powered effects of BuzzStream, Kayla immediately switches the background to an interactive educational wallpaper reflecting the theory and elements related to it. This immerses the students more and makes the learning process enjoyable and more visual.

5. Key Technical Challenge

The principal technical challenge of our project is to optimize the integration of large machine learning models into real-time video streaming on edge devices. With their growth in complexity, these models have become increasingly demanding in computational power and memory allocation, hence resulting in a slower inference time and being unsuitable in a low-latency live stream setting.

This complexity arises in two essential areas:

1. The trade-off between model size and inference latency: Larger models, while potentially more precise, consume more computational and memory resources and increase the inference time, thus interrupting the smoothness and real-time interaction of video streams. Addressing this challenge requires the exploration and implementation of "lite" versions of these models that maintain an acceptable level of accuracy while being smaller and faster.
2. The accuracy vs. Inference latency dilemma: The trend of focusing on model accuracy often results in compromising the speed of inference, especially when ported into a system like a live stream video. The necessity for real-time interaction prompts for the need to balance the twins of accuracy and speed. Here, the precise challenge lies in the exploration of the tradeoff and achieving the right balance wherein the system performs with sub-second inference latency without conceding too much on the accuracy.

Our technical approach lies in developing an adaptive model framework where the model size is adaptable based on the Frames Per Second (FPS). This framework will operate with *small*, *medium*, and *large* versions of the models, adjusting in real-time to the available resources and the requirement for the speed of execution. Implementing such a dynamic model framework will thus address the key challenge of infusing large ML models into a real-time streaming setting without compromising on the real-time interaction and user experience.

BuzzStream aims to explore this design space of video streaming and real-time ML-based image processing systems, proceeding to engineer a solution that seamlessly strikes the right balance between complexity, accuracy, and inference speed.

6. Application Architecture

BuzzStream's architecture incorporates React.js for the front-end, Express.js for the back-end, and MongoDB for potential data management tasks. It also employs Tensorflow.js for "hot-swapping" different sizes of ML models in real-time, and AWS S3 for client-end adaptive fetching of diverse model sizes, thereby meeting sub-second latency requirements. This synergistic technology stack results in a robust, scalable, and interactive application.

More specifically, for real-time video streaming, the application will use the WebRTC API on the client and server sides. WebRTC's functionalities include live video transfer, which are essential to the core use-case of BuzzStream. Communication on the client-side will involve peer-to-peer video chatting, while server-side WebRTC communication will involve handling server-to-server communications where necessary.

For real-time ML inference, Tensorflow.js, a JavaScript library, will be used for running the machine learning models. Tensorflow.js enables the execution of ML models in-browser, thus catering to real-time video effects. It provides both pre-trained models ready for use and the ecosystem to train and deploy your models from scratch.

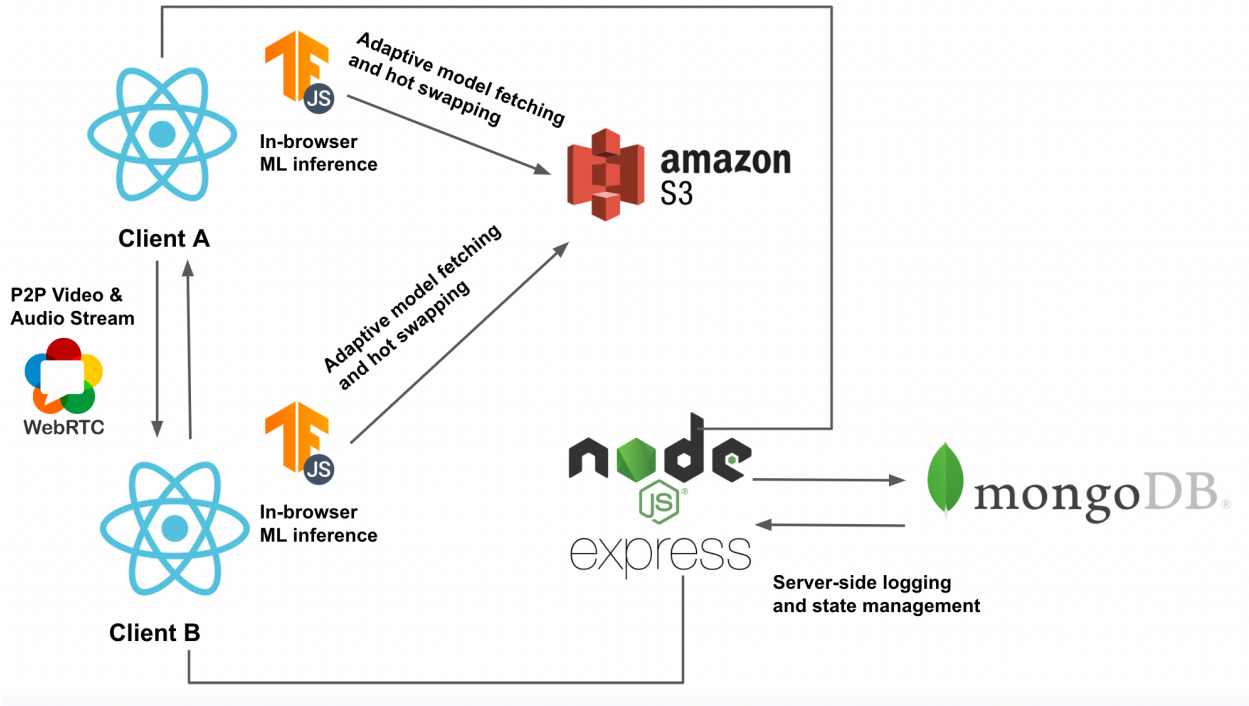

Figure: BuzzStream Application Architecture

7. Schedule and Skill Attainment

Week 1-2: MVP Video Live Streaming Web App

Brief: Begin by establishing a MVP focused on creating the base video live streaming web application. This stage includes setting up the basic front-end and back-end architecture, integrating WebRTC for live video streaming, and ensuring smooth video chat functionality.
Skill Attainment:

1. Learning experience of a perishable, but repeatable process: Learning to integrate the WebRTC for live video streaming into a web application. This newly learned process can be repeated for other video integrated applications.
2. Trend recognition: Spotting the trend of real-time interactions facilitated by video live streaming and understanding technology stacks used in modern video live web applications.

Week 3-5: Building and Optimizing Models

Brief: Focus on preparing the small, medium, and large-sized machine learning models. Subsequently, optimize these models to be browser-compatible and ready for real-time inference.
Skill Attainment:

1. Learning experience of a perishable, but repeatable process: The process of building and optimizing machine learning models is a perishable skill that can be transferred to future projects.

Week 6-7: Develop Adaptive Mechanism for Model Switching

Brief: At this stage, develop an adaptive algorithm capable of dynamically hot-swapping models in real-time. This crucial feature enables switching between models of different sizes when the Frames per Second (FPS) dips below a set threshold. This ensures maintenance of the desired latency and streaming quality.
Skill Attainment:

1. Learning experience of a perishable, but repeatable process: Creating algorithms for model switching is a repeatable process and proving to be a metaskill.
2. Trend recognition: This represents a key trend that optimizes performance of video streaming and machine learning applications in development.

Week 8: Testing and Debugging

Brief: Conduct comprehensive testing of the application, identifying potential bugs or areas of improvement, making necessary adjustments, and ensuring its overall effectiveness. The primary focus will be on testing model switching and its impact on latency and video stream quality.
Skill Attainment:

1. Distinction between facts and fiction/opinions: Understanding and using different testing methodologies helps discern hard facts about app performance from subjective opinions.

Week 9: Final Testing and Report

Brief: Perform a final round of comprehensive testing to ensure the efficacy of all features of the web app. Following this, write a report to summarize the key learnings from this project.

Skill Attainment:

1. Distinction between facts and fiction/opinions: Evaluating and summarizing project outcome helps in separating productive results (facts) from initial assumptions or beliefs (opinions).

	Week 1	**Week 2**	**Week 3**	**Week 4**	**Week 5**	**Week 6**	**Week 7**	**Week 8**	**Week 9**
Task	MVP	MVP	Model	Model	Model	Integrate	Integrate	Testing	Report
Huan	☑️	☑️	☑️	☑️	☑️	☑️			☑️
Gabriel	☑️	☑️				☑️	☑️	☑️	☑️

Figure: Table of Timeline and Workload Balancing

8. References

Engdahl, S. (2008). *Back to basics: HTTP video streaming*. Amazon. https://aws.amazon.com/blogs/media/back-to-basics-http-video-streaming/

Laukens, N. (n.d.). Adaptive Streaming - A brief tutorial - EBU. https://tech.ebu.ch/docs/techreview/trev_2011-Q1_adaptive-streaming_laukens.pdf

Moreira, L. (n.d.). *Leandromoreira/A hands-on introduction to video technology: Image, video, codec (AV1, VP9, H265) and more (ffmpeg encoding)*. GitHub. https://github.com/leandromoreira/digital_video_introduction

Mühler, V. (n.d.). *Justadudewhohacks/face-api.js: JavaScript API for face detection and face recognition in the browser and nodejs with tensorflow.js*. GitHub. https://github.com/justadudewhohacks/face-api.js

Tensorflow. (n.d.). *Tensorflow/tfjs: A webgl accelerated javascript library for training and deploying ML models.* GitHub. https://github.com/tensorflow/tfjs

WebRTC For The Curious. WebRTC for the Curious. (n.d.). https://webrtcforthecurious.com/