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

Gabriel A. Lopez Lugo, Huan Xu

1. Motivation

In the last few decades, data generation has increased at an exponential rate, so fast that network performance is currently bottlenecked by bandwidth. Server-side computing is no longer feasible in many scenarios, such as with self-driving vehicles, autonomous robots, etc. Hence, new trends have been moving towards an edge computing future, in which devices at the edge of the network process data as close to the source and only communicate with the server when needed. However, there are challenges in regards to the heterogeneity of edge devices since not all are built equally, meaning that the runtime system load will change dynamically. Moreover, with the advent of AI in most applications, modern devices require heterogenous resources (e.g., GPU, AI Accelerators) to properly support machine learning inference. This new collaboration between edge and cloud points towards a demand for low-latency applications. Yet, there is a lack of open-source test benches for latency-accuracy tradeoff explorations of machine learning inference on edge. For this project, we aim to build such a test bench for edge ML inference in the video streaming domain.

2. Introduction

The COVID-19 pandemic sparked an unprecedented surge in online interactions, creating a demand for sophisticated and interactive online meeting software. As a response, technology titans like Zoom, Microsoft Teams, and Google Meet introduced real-time video streaming software. Nevertheless, real-time video streaming is widely used post-pandemic while being 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. Resource contention becomes a considerable hurdle as both codec and ML inference significantly utilize GPU. For this reason, we believe that video streaming can benefit from edge-cloud collaboration to reduce latency, as the edge device can apply any background removal before sending it to the cloud, essentially saving server processing time.

In this project, we aim to explore the design space of video streaming and ML-based image processing systems. We developed “BuzzStream,” a video chat web application, employing innovative technologies such as WebRTC and Paddlepaddle.js. Our goal is not only to address the issues mentioned above but also to provide a lightweight test bench that can leverage custom, non-proprietary models to achieve seamless live streams blended with AI image processing features.

3. 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.

3.1 Closed-sourced Solutions

Zoom, as one of the leading platforms for video conferencing, has shown an state-of-the-art 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. They work great as a commercial solution, but we aim to explore the design space of video streaming and ML-based image processing systems using non-propritory, open-source models, and these commercial platforms do not allow conducting experiments using customized machine learning models.

3.2 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, existing open-source streaming solutions do not have built-in support for ML inference in browser, and require significant modifications to conduct experiments for latency-accuracy explorations of ML inference.

3.3 Open-source ML-based Video Processing Solutions

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 paddle.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.

3.4 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 paddle.js, which enable 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. It is worth noting that we do not aim to compete with existing, commercial video steaming solutions, as they are tailored to fit customers' specific needs and backed by abundant cloud compute and storage resources. Instead, we aim to develop a lightweight, modular, and fully open-source test bench for latency-accuracy explorations in edge + ML inference, specifically in video streaming domain.

4. Features

BuzzStream is a dynamic web-based application that provides a platform for users to create, participate in, and enjoy engaging video streaming-based interaction. Here are some key features that BuzzStream provides:

4.1 Interactive Video Chat

Within BuzzStream, users can call other users to join a video chat where they interchange camera feed and audio. The purpose of the video chat feature is to test our performance-adaptive model switching algorithm with a real-world application.

4.2 AI-Based Video Stream Enhancement Tools

The application incorporated AI-powered effects into video streams. Specifically, this project looks at the problem of background removal using a deep-learning based human segmentation model. This feature allow users to manipulate their backgrounds and replace non-human part with a static background. Background removal is a common use case for ML + video streaming applications, and we want to study the latency-accuracy tradeoff for this application as long as automate the model size heterogeneity using our adaptive switching algorithm.

4.3 Performance-adaptive Model Switching

To accommadate for edge devices of varying hardware capacity and dynamic runtime load, the application uses an adaptive model algorithm that smoothly transitions between a small, medium, and large version of the model depending on the current Frames Per Second (FPS) of the stream. The framework leverages a increasing confidence value to determine if it should change to a larger version of the model. If the FPS goes below a threshold, the framework will switch back to a smaller version with confidence value increasing a slower rate.

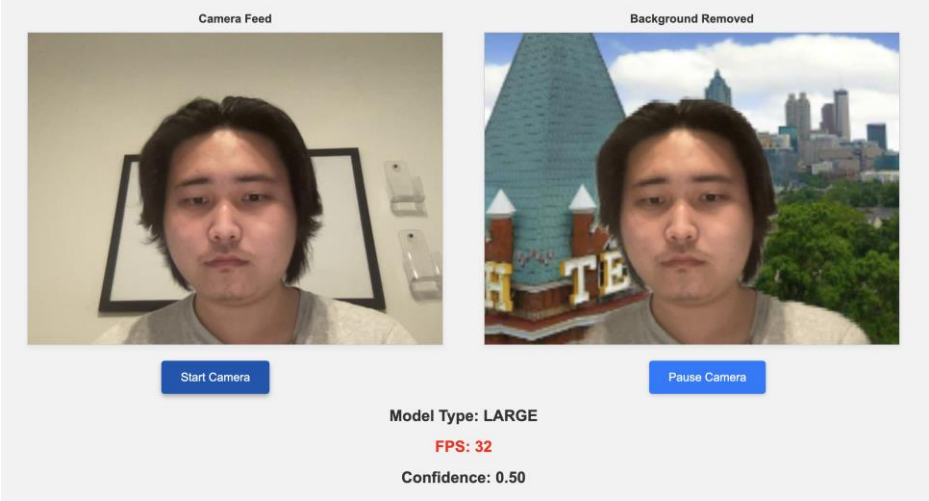

Figure 1: Large model with high accuracy, yet low FPS

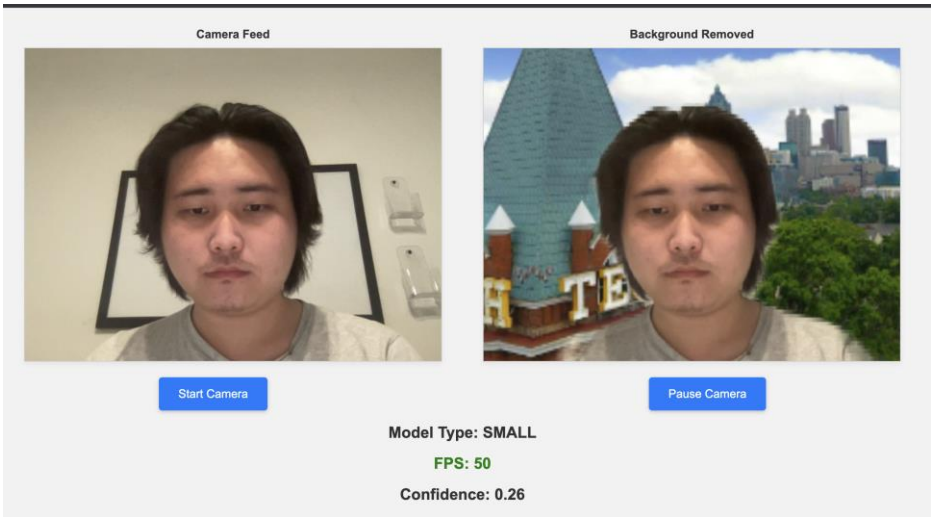

Figure 2: App switches to small model for better FPS at the cost of image quality

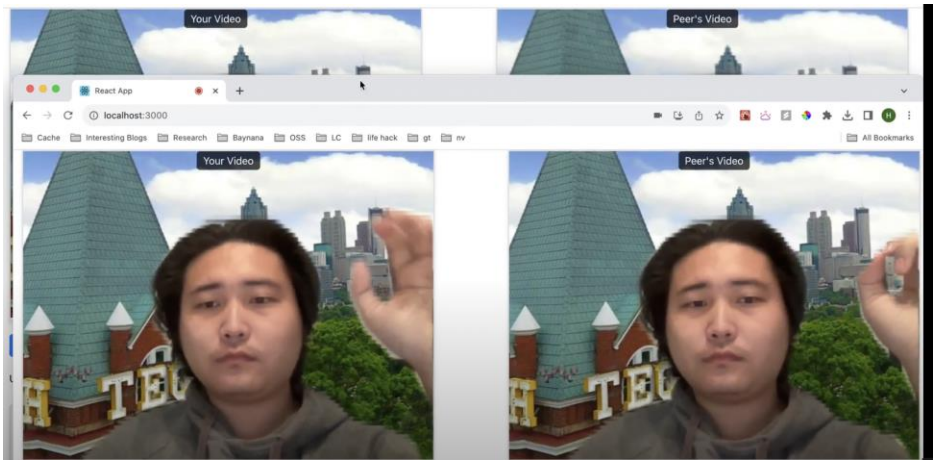

Figure 3: Peer-to-peer video streaming with adaptive model switching

4.4 Lightweight Test Bench

BuzzStream is foremost a test bench for exploring latency-accuracy tradeoffs of real-time model inference on edge edvices. Thus, the application is fast and lightweight with the capability of accepting custom models compatible with Paddlepaddle.js, and therefore Tensorflow or ONNX.

5. Application Architecture

BuzzStream's architecture incorporates React.js for the front-end, Express.js for the back-end. It also employs Paddlepaddle.js for “hot-swapping” different sizes of ML models in real-time by pulling from the Model Zoo, thereby meeting sub-second latency requirements. This synergistic technology stack results in a modular and robust 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, Paddlepaddle.js, a JavaScript library, will be used for running the machine learning models. Paddlepaddle.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.

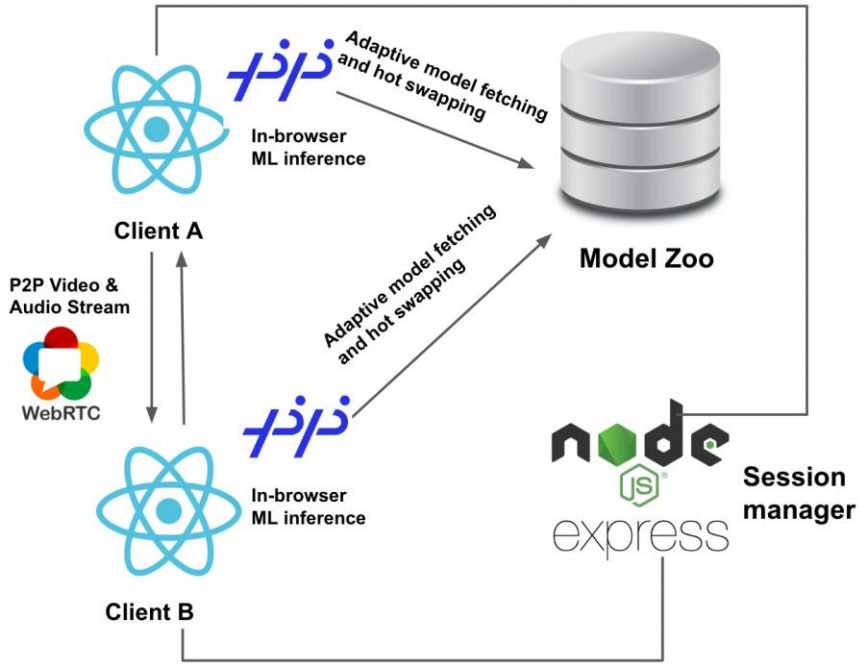

Figure 4: BuzzStream Application Architecture

6. Future Work

A continuation of our program would incur exploration of other real-time video streaming settings. One area of interest is Cloud Gaming, which has public audience appeal yet not much demand due its high subscription costs. AI solutions in gaming have become popular, such as Nvidia's Deep Learning Super-Sampling (DLSS) and AMD's FidelityFX Super Resolution (FSR), models that upscale lower resolution frames to allow for better gaming performance and power efficiency. We believe that the same methodology can be levaraged for Cloud Streaming to maintain respectable latency during gaming while reducing subscription fees and decreased server-side power consumption.

We plan on connecting BuzzStream to a database, especifically Amazon's AWS S3 because of its ease of use, to aggregate and collect model inference analytical data as model properties to best match the best-performing models to similar harward specs. Furthermore, our application is hardcoded to only three models pre-labeled with either small, medium, or large (one-dimensional view), when in reality models have multiple dimensional views (e.g., quantization level, estimated power consumption) that could be used for distinct cases within edge computing. Ideally, the user can use the AWS S3 to upload their own custom models and the application can then download models based on their propertie

7. Deliverables

Final Presentation and Report

Inference Engine

- The inference does not have a demo, but it is used in both the BuzzStream Adapative and BuzzStream Video demos to run background removal via human segmentation.

BuzzStream Adaptive Demo

- Demo of the BuzzStream application, which accepts the camera feed and removes the background from the video feed. The application automatically switches to a smaller model if the FPS drops below a certain threshold. Additonally, it will also automatically switch back to a larger model if it is confident enough that the FPS will remain high.

BuzzStream Video Chat Demo

- BuzzStream Demo Video Chat allows users to video chat with each other while removing the background from the video feed. Notably, the application will detect the current FPS and automatically switch to a smaller model if the FPS drops below a certain threshold. Beyond that, the application will also automatically switch back to a larger model if it is confident that the FPS will be high enough.

- The application has two parts, the client and the server. The client is a React application that uses the webcam to capture video and sends it to each other via WebRTC. The client also removes background using our inference engine buzzstream/humanseg. The server is a Node.js application that manages user states.

8. Skills Attainment

- **Trend Recognition:**
    - In developing the adaptive inference algorithms, we've aligned ourselves with the prevailing trend of real-time machine learning applications and edge computing. Recognizing and addressing the crucial equilibrium between computational efficiency and quality of output represents a significant skill advancement. This awareness becomes particularly salient in the context of browser-based applications, which operate under strict resource limitations.
    - By doing literature review on the live streaming topic, we spotted the foundational changes for live streaming technologies, enabling lower and lower streaming latency. The live streaming technologies evolved from HLS (60 seconds latency) to DASH (10 seconds latency) to RTMP (1 second latency), and finally to RTP and WebRTC (under 500 ms), enabling more and more interactive applications. New capabilities of the video live streaming technologies introduced new trade-offs for ML-assisted live-streaming applications: the end-to-end application latency is shifting to ML inference latency bound rather than streaming latency bound. Applications are requesting novel technical solutions to solve this new challenge.
- **Perishable Skills:**
    - Our development of the adaptive inference algorithm, which seamlessly toggles between the full-fledged and the distilled models to suit the dynamic user runtime environment, stands as a testament to the utility of this skill. The algorithm is a manifestation of our learning, aiming to refine the balance between accuracy and speed and thereby guaranteeing the model's robust performance under diverse computational loads and operational conditions, while achieving a better user experience.
    - During the last two weeks, we learned to convert machine learning models that are trained using PyTorch into the general ONNX format to be loaded into browser, which is a perishable skill that can be transferred to future projects. Given the fact that more and more machine learning inference is conducted on user’s device instead of servers, for example in federated learning setting, we recognize the importance of deploying machine learning models on edge settings like browser.

9. References

Chu, L., Liu, Y., Wu, Z., Tang, S., Chen, G., Hao, Y., Peng, J., Yu, Z., Chen, Z., Lai, B., & Xiong, H. (2022). PP-HumanSeg: Connectivity-Aware Portrait Segmentation With a Large-Scale Teleconferencing Video Dataset. Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV) Workshops, 202-209. GitHub. https://github.com/PaddlePaddle/PaddleSeg/blob/release/2.8/contrib/PP-HumanSeg/paper.md

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

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