LevelUp: A thin-cloud approach to game livestreaming

Landon P. Cox
Microsoft Research

Lixiang Ao
University of California, San Diego

***Abstract*—Game livestreaming is hugely popular and growing. Each month, Twitch hosts over two million unique broadcasters with a collective audience of 140 million unique viewers. Despite its success, livestreaming services are costly to run. AWS and Azure both charge hundreds of dollars to encode 100 hours of multi-bitrate video, and potentially thousands each month to transfer the video data of one gamer to a relatively small audience.**

In this work, we demonstrate that mobile edge devices are ready to play a more central role in multi-bitrate livestreaming. In particular, we explore a new strategy for game livestreaming that we call a thin-cloud approach. Under a thin-cloud approach, livestreaming services rely on commodity web infrastructure to store and distribute video content and leverage hardware acceleration on edge devices to transcode video and boost the video quality of low-bitrate streams. We have built a prototype system called LevelUp that embodies the thin-cloud approach, and using our prototype we demonstrate that mobile hardware acceleration can support realtime video transcoding and significantly boost the quality of low-bitrate video through a machine-learning technique called super resolution. We show that super-resolution can improve the visual quality of low-resolution game streams by up to 88% while requiring approximately half the bandwidth of higher-bitrate streams. Finally, energy experiments show that LevelUp clients consume only 5% of their battery capacity watching 30 minutes of video.

I. Introduction

Game livestreaming services are large and growing. Between 2017 and 2019 Twitch increased the average number of concurrent viewers from nearly 750,000 to over 1.2 million and doubled the average number of concurrent streams from 25,000 to 50,000[1]. Yet despite despite the popularity and scale of services like Twitch and Facebook Live, streaming live multi-bitrate video remains expensive.

Multi-bitrate livestreaming services typically accept video from a broadcaster at a single bitrate and transcode the video at multiple bitrates so that viewers can adapt their stream quality to network conditions. When network conditions are good clients stream high-quality video, and as network conditions worsen clients stream lower-quality video. Generating multiple video qualities in realtime is computationally intensive and requires specialized hardware like graphical processing units (GPUs) or dedicated hardware transcoders. Unlike commodity cloud infrastructure like web front-ends, blob storage, and content distribution networks (CDNs), video transcoding has not achieved the economies of scale needed to drive down costs.

[1] twitchtracker.com/statistics

As a result, for 100 hours of live multi-bitrate video, AWS and Azure can charge hundreds of dollars for transcoding and thousands of dollars for data transfer. In comparison, third-party service Wowza charges less than $20 to livestream 100 hours of single-bitrate video. This makes livestreaming multi-bitrate video cost-prohibitive for small companies and a significant savings opportunity for large companies like Amazon, Facebook, and Microsoft.

In this paper, we observe that edge devices are ready to play a more central role in multi-bitrate livestreaming. We propose a new approach to livestreaming based on the notion of a *thin cloud*, in which the cloud provides commodity storage and CDN infrastructure, but does not supply expensive GPUs and other hardware accelerators. We have implemented a prototype system called *LevelUp* that embodies this approach. LevelUp is a game livestreaming service with refactored responsibilities. The cloud is responsible for storing and distributing video segments and other files, and edge devices are responsible for multi-bitrate transcoding and boost the quality of low-bitrate streams.

Our approach is enabled by the rapid deployment of hardware acceleration on the edge. Because gaming, photography, and media playback are critical smartphone applications, GPUs and video-transcoding hardware are standard features of smartphone chips. Furthermore, the next wave of accelerators target machine learning (ML) and are deployed on tens of millions of smartphones. These accelerators handle complex ML workloads and will become as commonplace as GPUs.

LevelUp uses edge hardware acceleration to reduce cloud costs in two ways. First, it uses broadcasting devices' hardware video encoders to generate and upload videos at different resolutions. This obviates the need to transcode in the cloud. Second, LevelUp viewers use a hardware-accelerated convolutional neural network (CNN) to improve the visual quality of lower bitrate video. This can significantly reduce the amount of data the cloud transfers to clients without sacrificing visual quality.

We do not claim any contributions to ML. Work on compressing models to run more effectively on mobile devices and constructing models that achieve better inference and/or performance are orthogonal to LevelUp. Instead, our contributions are as follows:

- We identify multi-bitrate transcoding and data transfer as major costs for multi-bitrate livestreaming, and propose a thin-cloud approach to reducing those costs.

- We identify super-resolution as a way to mitigate the high cost of transferring game-stream data. In particular, we observe that super-resolution models trained on specific game content can provide additional quality improvements over networks trained on different games.
- Using our LevelUp prototype, we show that realtime multi-bitrate transcoding is feasible on commodity devices, and that smartphones can perform realtime super-resolution of reduced-resolution videos. We further demonstrate that super-resolution can improve the visual quality of reduced-resolution game streams by up to 88%, and that training a super-resolution model on specific game content can improve visual quality by over 25% compared to a model trained on a diffferent game's content. Finally, energy experiments show that LevelUp clients spend only 5% of their total battery capacity to super-resolve 30 minutes of low-resolution video.

The rest of the paper is organized as follows: Section II provides background information on livestreaming, mobile-device hardware, and image-similarity metrics; Section III articulates the design principles underlying LevelUp, Section IV describes the LevelUp design and implementation, Section V describes our experimental evaluation, Section VI describes related work, and Section VII provides our conclusions.

II. Background

In this Section, we provide background information on video streaming, mobile hardware accelerators, and image-similarity metrics.

A. Live streaming

The two most widely used live-streaming protocols are Real-Time Messaging Protocol (RTMP) and HTTP Live Streaming (HLS). RTMP sits directly above TCP. RTMP broadcasters split streams into small audio and video chunks (e.g., 4KB videos) and send them over a persistent TCP connection to an RTMP server. Each stream viewer maintains a persistent TCP connection to the RTMP server, and the RTMP server pushes new chunks to viewers as they become available. In its simplest form, an RTMP server acts as a pass-through relay for media chunks from broadcasters to viewers. RTMP also supports multi-bitrate coding, in which an RTMP server transcodes incoming media into different qualities (e.g., resolutions) before forwarding chunks to viewers. In this mode, viewers communicate their desired level of quality to the server.

The primary benefit of RTMP is that it provides low end-to-end latency. Servers can forward chunks as soon as they are available because RTMP pushes media over persistent TCP connections. This is much faster than viewers' polling for new chunks, or worse, initiating a new connection for each download. In addition, each chunk in an RTMP stream contains a brief playback period. For example, if a broadcaster uploads at 250kbs and the server expects 4KB chunks, then each 4KB chunk will capture 128ms of video. Thus, a broadcaster can send new data to the RTMP server every 128ms.

Despite its low latency, RTMP requires dedicated servers that do not easily integrate CDNs. Because of this, most streaming services use HLS and Dynamic Adaptive Streaming over HTTP (MPEG-DASH), which scale better than RTMP but sacrifice end-to-end latency. Like RTMP, HLS divides video streams into segments, but because it runs on top of HTTP segments are stored as normal files (e.g., a self-contained H264 video) and viewers retrieve new segments by issuing HTTP get requests. HTTP/2 supports server-side push but is not widely deployed or integrated with HLS.

HLS segments are large relative to RTMP chunks to amortize the cost of HTTP. HLS segments typically capture 2-10 seconds of playback, and these longer lengths create additional end-to-end delay. For example, if a stream uses 10-second segments, then a broadcaster must wait 10 seconds before uploading the first segment, and stream viewers will always be more than 10 seconds behind the broadcaster. For many applications, seconds of delay is tolerable. For example, celebrities on Facebook Live and popular on Twitch attract large enough audiences (e.g., thousands if not millions of viewers) that interactivity is not practical. Lower latency is always better, but RTMP is often overkill for broadcasts that scale beyond dozens of participants.

For a viewer to retrieve segments over HTTP, it must know segments' URLs. Thus, each HLS stream contains a plaintext playlist that describes the URLs for all of a stream's segments. This playlist also helps with multi-bitrate streaming by describing where viewers can download segments of different qualities. Viewers adapt the quality of their stream by downloading the segments that best match their network conditions.

HLS and MPEG-DASH scale well because CDNs can easily distribute playlists and video segments. This is the main reason that all major streaming services support MPEG-DASH and/or HLS, including Facebook Live, Twitch, Netflix, and YouTube. LevelUp's aim is to repurpose as much web infrastructure as possible, and it also uses HTTP for transport. However, HTTP is compatible with commodity web infrastructure, but multi-bitrate streaming still requires special-purpose machinery. For both RTMP and HLS, broadcasters upload video at the best bitrate they can, and a specialized server transcodes video segments into different qualities.

Transcoding live video in the cloud is expensive. Transcoding 100 hours of live video at resolutions of 1920x1080 and below at 30 frames-per-second (FPS) using Azure Media Services costs over $300 (not including transfer costs).[2]. Amazon's Media Live product costs over $500 per month, per video channel at 1920x1080 resolution and at 30 FPS, also not including data transfer costs[3]. A primary reason that multi-bitrate transcoding in the cloud is so expensive is that transcoding in realtime requires hardware acceleration, such as GPUs or other dedicated transcoders. In addition, the cost to deliver high-quality videos to clients at even moderate scale

[2]As of 12-2019: azure.microsoft.com/en-us/pricing/details/media-services/
[3]As of 12-2019: aws.amazon.com/medialive/pricing/

can be expensive. Azure charges between $5,000 and $7,500 to transfer 150-500 TB/month. AWS Media Connect prices are similar.

We believe that mobile devices can play an important role to play in reducing these costs. LevelUp explores this idea by using smartphone hardware on the edge to (1) eliminate cloud-based transcoding, and (2) improve the visual quality of reduced-bitrate video.

B. Mobile hardware accelerators

System-on-chip (SoC) and smartphone manufacturers deploy hardware accelerators to meet the computational demands of mobile applications. As it became clear that gaming was important for smartphones, devices of all qualities began to ship with GPUs. Similarly, as smartphones became users' primary device for taking photos and videos, SoCs integrated dedicated hardware for video encoding and decoding. For example, iPhones have included hardware for encoding and decoding H264 video for many generations, and they have included hardware for encoding and decoding H265 video since the iPhone 6.

The next wave of SoC accelerators target machine learning (ML) through on-chip ML co-processors. For example, Apple added a two-core ML accelerator called a Neural Engine to the A11 processor included in the iPhone 8, iPhone 8 Plus, and iPhone X. Apple increased the number of cores in this accelerator to eight in the A12 and A13 processor. The Qualcomm Snapdragon 855 used by Google's Pixel 4 and other high-end Android smartphones, also supports hardware-accelerated neural-network execution.

This advanced hardware is much more widely deployed on smartphones than in public clouds. As of May 2018, there were an estimated 118.7 million active iPhone 7 devices, 118.5 iPhone 6s devices, over 60 million iPhone 8 devices, and over 40 million iPhone X devices [3]. In the first quarter of 2020, Apples sold an estimated 32 million devices with its A13 chip [4]. Thus, just among active iPhones there are hundreds of millions of processors capable of hardware-accelerated video transcoding. While individual computational units in the cloud may be more capable than their mobile counterparts due to power constraints, the cloud will never reach the same scale of deployment collectively achieved by smartphones. Furthermore, new hardware accelerators, such as ML processors, will be deployed at scale far more rapidly on mobile devices through natural user upgrade cycles than is practical for public-cloud providers such as AWS and Azure.

LevelUp uses these trends to drive down the cost of livestreaming. In particular, as long as smartphones are capable of transcoding video and performing ML-based image processing in realtime, then it will be more cost effective to shift workloads to devices on the edge than solely relying on cloud infrastructure.

C. Image-quality metrics

To properly evaluate whether LevelUp can meaningfully boost the quality of low-bitrate videos, one needs an image-quality metric. In particular, one must have a way to characterize how well a transformed image (e.g., by video encoding and decoding or by ML processing) approximates the original. Image quality is subjective, and image-quality metrics try to predict this subjectivity. For example, image-quality metrics typically only consider differences of luminance because studies have found that humans react more negatively to loss of luminance than loss of chrominance.

Peak-signal-to-noise-ratio (PSNR) is a simple and well known image-similarity metric. PSNR is calculated by averaging the pixel-by-pixel mean-squared-error (MSE) of the luminance (Y) and chrominance (Cr and Cb) channels of two images. PSNR has known limitations, and researchers proposed the Structural Similarity Index (SSIM) as an alternative to PSNR over 15 years ago [21]. SSIM is more complex than PSNR and tries to capture the perceived change in structural information caused by an image transformation. SSIM is generally viewed as an improvement over PSNR, but both fail to consistently reflect human perception [1]. Of particular concern for LevelUp, measuring the quality of individual frames may not capture video quality.

Netflix's Video Multi-method Assessment Fusion (VMAF) is a relatively new quality metric designed specifically for video [1]. VMAF uses ML to model how humans grade the quality of a sequence of images. VMAF combines several metrics, including Visual Information Fidelity (VIF) [18], Detail Loss Metric (DLM) [15], and luminance differences between adjacent frames to measure motion fidelity. VMAF uses a support vector machine (SVM) regressor to assign weights to these elementary metrics by training it with data from user studies. By doing so, VMAF tries to avoid the weaknesses of individual metrics and better approximate how a human would evaluate a video's quality. We primarily rely on VMAF to evaluate LevelUp.

III. Design principles

To reduce the cost of game livestreaming, we designed LevelUp using the following principles.

Transcode videos on edge devices. As discussed in Section II-A, existing cloud services charge an order of magnitude more for multi-bitrate livestreaming than for single-bitrate livestreaming. And as discussed in Section II-B, nearly all mobile devices possess dedicated hardware for encoding and decoding video. Thus, LevelUp broadcasters directly transcode their game output on their smartphones in realtime and upload these video files to the cloud for distribution. Our LevelUp prototype encodes gameplay at three resolutions: high (1920x1080), medium (854x480), and low (480x270).

Even with the impressive capabilities of commodity devices, shifting work from the cloud to smartphones raises the question of whether these additional responsibilities will be too demanding. In particular, LevelUp's approach asks clients to expend additional energy and network bandwidth to create and upload multiple videos.

For energy, we strongly suspect that the marginal energy impact of multi-bitrate coding while playing a game is small.

Gaming is already very energy intensive, requiring a device's screen, CPU cores, and GPU to be in high-power states. On top of that, game streaming via services like Twitch or Mobcrush keeps a device's network radio in a high-power state and uses the device's hardware video-encoder to generate single-bitrate videos. LevelUp clients' energy usage above a service like Twitch or Mobcrush would be due to encoding and uploading multiple videos instead of one.

Fortunately, compared to other tasks involved in game streaming, hardware video-encoding is one of the most energy efficient. Dedicated hardware is far more efficient at transcoding than performing the same task on a CPU or GPU. Furthermore, the additional effort of generating and uploading medium- and low-resolution videos is small compared to high-resolution videos. For example, our results in Section V-A3 show that medium-resolution game streams are an order of magnitude smaller than high-resolution streams.

Nonetheless, some broadcasters may find that encoding and uploading high-, medium-, and low-resolution videos is too resource intensive. This could be due to transient periods of depleted battery or poor connectivity, or may be due to persistent issues like a network data cap. Platforms may also wish to limit their egress bandwidth by serving lower-bitrate streams. LevelUp mitigates the effect of these constraints through hardware-accelerated ML on viewers' devices.

Boost video quality with ML. Transcoding videos on a broadcaster's device instead of the cloud will reduce the cost of multi-bitrate streaming, but it asks broadcasters to create and upload more videos. In some cases broadcasters may save resources by encoding and uploading only reduced-resolution videos. Generating reduced-resolution videos uses less energy than generating high-resolution videos, and based on our results in Section V-A3, uploading medium- and low-resolution videos would save significant bandwidth compared to uploading all three qualities. Transferring lower bitrate video also offers large potential savings for platforms. Of course, reduced-resolution videos look worse than high-resolution videos. But like LevelUp broadcasters, LevelUp viewers have sophisticated hardware accelerators, and they can use these accelerators to mitigate the diminished visual quality of low-bitrate video.

Single-image super-resolution is an ML approach to improving the quality of reduced-resolution images [11], [22]. Unlike a simple interpolation, super resolution aims to model the complex, non-linear mapping of low-resolution image representations to high-resolution representations. That is, a super-resolution algorithm takes a low-resolution image and outputs an estimate of the image's high-resolution version. Initial work on super resolution used sparse-coding techniques [8], [23], and more recent work has investigated training neural networks [7], [14], [16], [19].

LevelUp uses the convolutional neural network (CNN) for super resolution described in [19]. Even though this CNN is relatively lightweight (it has just four layers), running it on a CPU would be too slow for livestreaming. Instead, LevelUp viewers run the CNN on their devices' ML co-processors. At the moment, these accelerators are available on smartphones like recent iPhones and Google Pixels, but this hardware will trickle down to all device levels in the near future.

LevelUp is not the first system to use super-resolution for video streaming. NAS [24] also trains a super-resolution deep neural network (DNN) to learn the mappings from low-quality to high-quality videos. NAS clients download and use a large DNN to transform lower-quality frames into higher quality. NAS uses a heavier-weight neural net than LevelUp, and as a result NAS models are over 100MB (LevelUp's are hundreds of KBs) and NAS must run on a desktop-class GPU.

More fundamentally, NAS targets pre-recorded content, and trains its DNNs on the same videos that clients stream. This approach will not work for livestreaming, because streamed content is not known in advance. Dejavu [12] leverages similar insights and techniques to enhance video conferencing. We hypothesize that games' visual content presents a major opportunity to use super-resolution for livestreaming.

Train a different model for each game. While many livestreams may change from broadcast to broadcast, a potential advantage of super-resolving gaming content is that there is significant visual similarity across game sessions. How a game's characters and objects interact with a game setting will change with each session, but the visual elements of those characters, objects, and settings will be consistent. Thus, LevelUp trains CNNs with video data from prior game sessions.

These CNNs learn general rules for upscaling arbitrary images, and they learn how specific visual elements within a game should be upscaled. This approach places LevelUp between neural nets that are trained to upscale arbitrary content and those that are trained to upscale very specific content (e.g., NAS).

LevelUp can offer CNNs to viewers that are tuned to the specific game content that they are streaming. For example, for games that have different levels or stages, a viewer might apply a different CNN to each level or stage of the game. Alternatively, for combat games like Super Smash Brothers or FortNite, a viewer might receive a CNN that has been trained to super-resolve the specific set of characters. There are far more possibilities to explore than can be covered in this paper. Could a CNN be trained for a region within the geography of a game? Could a client use object-detection ML to identify which objects or characters are present in a game and adaptively apply the appropriate CNN? How often should a client switch CNNs? LiveNAS [13] recently investigated some of these questions and demonstrated that enlisting broadcasters to help maintain super-resolution models can be beneficial for live video, including game content. For the purposes of this paper, we put the questions of where models come from and how they are maintained to the side. LevelUp could rely on game developers or streaming services to provide a library of CNNs or it could adopt the client-assisted techniques proposed by LiveNAS.

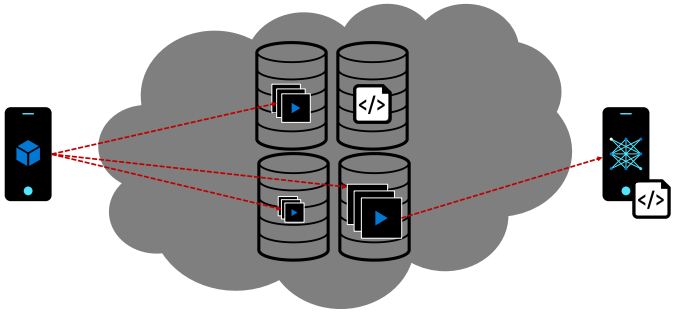

Fig. 1. High-level view of the LevelUp architecture. A broadcasting device uses its local hardware video-encoder to generate high-, medium-, and low-resolution videos. It uploads these files to cloud storage, where they can be distributed to viewers via CDN. Viewers use a stream playlist to identify the locations of each resolution stream, as well as the appropriate super-resolution model to apply to the stream it is viewing.

IV. Design and Implementation

The three main components of LevelUp's architecture are the broadcaster, viewer, and server. Figure 1 shows each of these components. Our LevelUp implementation is written in Objective-C for iOS. It takes advantage of several iOS frameworks, including CoreML and ReplayKit, and overall contains roughly 2,500 non-comment lines of code.

A. Broadcaster

The broadcaster is responsible for capturing video frames from the display as a user plays a mobile game, enqueuing those frames for the hardware encoder, and uploading video segments to the server as they complete.

To capture video frames from the display while a user plays a game, LevelUp uses the ReplayKit framework for iOS. ReplayKit allows third parties such as LevelUp to record audio and video in the background. To register with the ReplayKit system, an app must implement a user interface for handling remote account setup or registration, and a live-broadcast extension for processing video frames and audio clips. After registration and setup, a user can start streaming by selecting LevelUp as the destination screen recorder.

iOS imposes strict resource restrictions on live-broadcast extensions so that they do not compete for resources with the foreground application. In particular, it is critical that LevelUp stay under the 50MB memory limit and not compete with the streamed game for compute resources like the CPU and GPU. Fortunately, the ReplayKit framework is designed to make this straightforward, and LevelUp does not require significant compute resources besides the hardware video encoder.

The LevelUp extension is responsible for converting frames into video segments and then uploading those segments. Our current prototype encodes two-second segments at 30FPS. Two-second segments are an optimization for latency at the expense of bandwidth. Streaming delay grows with segment length, but the encoder can often compress more effectively when segments are larger.

Nonetheless, whenever ReplayKit gives LevelUp a new video frame, the handler enqueues it with the hardware encoder at three resolutions (1920x1080, 960x540, and 480x270). The extension then increments a frame counter (modulo 60), and checks if the counter is equal to 60. If the counter is equal to 60, then the current frame must be the last frame of the segment.

When a segment is complete, the extension directs the hardware encoder to finalize all videos it has queued. Once those requests return, the extension must upload the new videos as well as a new playlist to reflect that new segments are available. Uploading the new playlist and segments would be too slow to perform synchronously, so those tasks are performed on background threads that have been pre-initialized with HTTP connections to the server. Critically, uploading an updated playlist cannot begin until all of the video segments it references have been successfully uploaded.

Our playlist format is a simple JSON object with fields describing the locations of high-, medium-, and low-quality segments, the most recent segment number, and the location of any super-resolution CNNs that could be applied to reduced-resolution segments. It is important to note that broadcasters are not responsible for training a game's CNNs. We assume that LevelUp administrators or game developers will provide these models.

B. Viewer

The viewer is responsible for downloading the highest resolution video segment that its bandwidth will allow, queuing video segments as they download, and potentially passing reduced-resolution segment frames through the a super-resolution CNN. To start a stream, the LevelUp viewer downloads the playlist for a stream, constructs the URL for the first segment at a particular quality, and schedules the download on a background thread. Once the download has completed, the background thread notifies another thread that the video is ready to display, and the segment is enqueued with the video player. Then the whole process repeats for the next segment.

The process is slightly more complicated when a frame must be super-resolved. In this case, each frame from the downloaded and decoded video segment must first be converted to a single-channel grayscale image so that it can be input the LevelUp's CNN. This input representation allows the CNN to remain lightweight enough to process images in realtime on a mobile ML-accelerator. LevelUp uses iOS's CoreML framework for handling the CNN's input and output images, as well as scheduling tasks on the hardware accelerator.

Just as inputs to the CNN must be single-channel grayscale images, so too are the outputs of the CNN. Thus, before displaying a super-resolved frame, LevelUp must combine the grayscale output from the CNN with the original reduced-resolution image. This is done by using the output of the CNN (which is a full 1920x1080-resolution image) as the luminance channel (i.e., Y) for a new image. Then LevelUp uses bicubic interpolation to upscale the reduced-resolution frame to high-resolution, and uses the chrominance channels of this upscaled image (i.e., Cr and Cb) as the chrominance of the new, merged

image. Once the new image has been displayed, LevelUp looks to construct the next frame in the same manner.

We should note that our LevelUp prototype does not include logic for adapting stream quality to changing network conditions. This is an area of active research [17] and any reasonable implementation would be suitable as long as it does not compete for resources with the super-resolution CNN. Our prototype allows users to manually change their stream's quality, but this is obviously not ideal.

C. Server

The final component of the LevelUp architecture is the server. By design, the LevelUp server is extremely simple. Our current implementation is an Apache webserver with a php endpoint for accepting uploaded files. Of course, in a real deployment the server could be integrated with a CDN and other services for scaling HTTP workloads.

V. Evaluation

To evaluate LevelUp, we sought answers to the following questions:

- Can super resolution improve the visual quality of reduced-resolution game streams?
- Can mobile clients perform multi-bitrate coding in realtime?
- Can mobile clients super-resolve reduced-resolution video streams in realtime?
- What is LevelUp's energy overhead?

To answer the first question, we performed experiments with seven representative gaming videos from xiph.org[4]: Counter-Strike, Global Offensive (CSGO); Dota 2; Fallout 4; Grand Theft Auto V (GTAV); Rust; Starcraft 2; and The Witcher 3. Each 1920x1080-resolution video was 60 seconds long, and captured from Twitch at 60 FPS. The Rust video contained a single, continuous scene, and the other videos contained clips from different parts of a game. The xiph games included realtime-strategy games (DOTA2 and STARCRAFT2) and first-person action games (CSGO, Fallout4, GTAV, RUST, and WITCHER3). We used these videos and PyTorch to train, test, and validate a lightweight, convolutional neural network (CNN) with a well-known sub-pixel super-resolution architecture [19].

To answer the next two questions, we performed experiments with our prototype LevelUp implementation on five generations of Apple iPhones: an iPhone 11 Pro with an A13 processor and 4GB of RAM,, an iPhone Xs with an A12 processor and 4GB of RAM, an iPhone 8 with an A11 processor and 2GB of RAM, an iPhone 7 with an A10 processor and 2GB of RAM, and an iPhone 6s with an A9 processor and 2GB of RAM. Performing experiments with these devices allowed us characterize how well several generations of mobile processors ran LevelUp.

To answer the final question, we performed experiments with an iPhone Xs and 11 Pro using the popular iOS game Monument Valley. We viewed a stream with a full battery, and recorded the remaining battery capacity after 30 minutes.

All videos in our experiments were encoded using H264 for two reasons. First, unlike newer coding algorithms, such as AV1 [2], nearly all devices provide hardware-accelerated H264 decoding. Many mobile devices support hardware-accelerated H265 decoding (e.g., all iPhones since the iPhone 6), but H264 support remains much more common. Second, algorithms like H265 provide better compression than H264 but are often slower, which makes them less appropriate for game streaming. Though we limited ourselves to H264, we believe that LevelUp would behave similarly for any video coding algorithm that can be hardware-accelerated.

A. Super resolution

To test if a super-resolution CNN can provide better video quality than H264 for the same bitrate, we trained a CNN using gaming videos from xiph.org. Each video consisted of frames captured at a resolution of 1920x1080. For each video, we selected a four-second segment of 240 continuous frames, and randomly assigned frames to a testing and training set. When validating a model, we never included input frames from the model's testing or training sets. When validating a game model with frames from another game, we used all 3600 frames from the 60-second video. However, when validating a game model with frames from the same video, we excluded the testing and training frames, i.e., we used 3360 frames from the 54 seconds not used for testing and training.

We used ffmpeg to downscale and re-encode testing, training, and validation frames using ffmpeg at medium resolution (i.e., 854x480) and low resolution (i.e., 480x270). We re-encoded videos using ffmpeg's default H264 encoder (libx264) with a Constant Rate Factor (CRF) of 23 and tuned coding to low latency. In all cases, the target frame for an input frame was the corresponding, full-resolution frame from the original xiph video.

Testing, training, and validation input frames had less visual information than the original frames due to their reduced resolution and lossy H264 encoding. We used these decoded frames to train a CNN for each game with upscale factors of two and four. For an upscale factor of two, our input frames were 854x480, and each model generated a 1708x960 output image. To match the original resolution, we used bicubic interpolation to upscale the 1708x960 images to 1920x1080. For an upscale factor of four, our models took a 480x270 image as input and directly output a full-resolution, 1920x1080 image. We used PyTorch to train each model with loss rate of 0.0001 for 1000 epochs.

1) PSNR, SSIM, and VMAF image quality: We fed each game's CNN decoded frames from a validation set. For each medium- or low-resolution input frame, the CNNs output a high-resolution frame that we compared to the corresponding original high-resolution frame using PSNR, SSIM, and VMAF. Figure 2 shows how super resolution changed the quality of medium-resolution H264 videos, and Figure 3 shows how super resolution changed the quality of low-resolution H264

[4]https://media.xiph.org/video/derf/

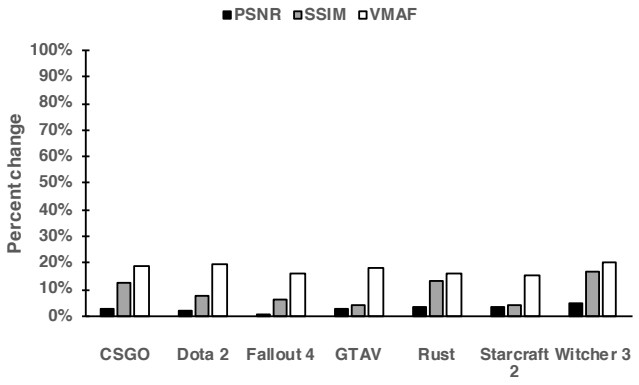

Fig. 2. Percent change in PSNR, SSIM, and VMAF after super resolution for 854x480 frames.

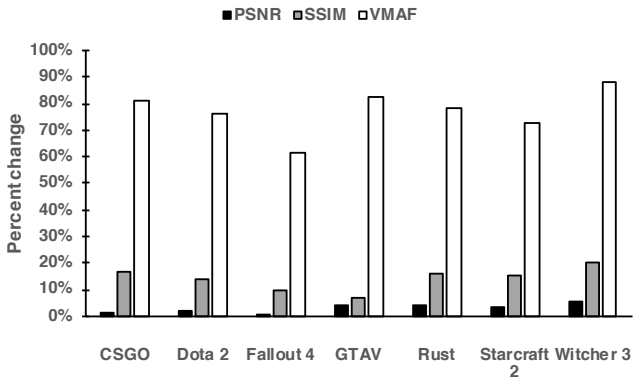

Fig. 3. Percent change in PSNR, SSIM, and VMAF after super resolution for 480x270 frames.

videos. For these experiments we used models trained and validated with frames from the same game.

The first result from these graphs is that super resolution affected VMAF far more than PSNR and SSIM. LevelUp barely changed the PSNR of any game stream at any resolution. LevelUp improved SSIM between 1 and 13% on medium-resolution inputs, and between four and 20% on low-resolution inputs. On the other hand, LevelUp improved the VMAF of medium-resolution frames between 15.1% (Starcraft 2) and 20.5% (Witcher 3). Even more impressive, LevelUp improved the VMAF of low-resolution frames between 61.2% (Fallout 4) and 88.4% (Witcher 3).

In Section II-C we discussed differences among PSNR, SSIM, and VMAF, but to gain a better sense why LevelUp improves VMAF scores more than PSNR and SSIM, consider the examples in Figure 4 and Figure 5. Figure 4 shows a 200x200 detail from the game Dota 2, and Figure 5 shows a 200x200 detail from the game Witcher 3. Moving left to right, the figures show a game detail at full resolution, medium resolution, super-resolved medium-resolution, low resolution, and super-resolved low-resolution. For both games, super resolution qualitatively improves edge sharpness. The medium- and low-resolution frames are blurry because of interpolation, whereas the super-resolved frames preserve more of the original's details, such as teeth, wisps of hair, and folds of clothing.

Fig. 4. Example 200x200 detail from Dota 2 at high resolution (1920x1080), medium resolution (854x480), super-resolved medium resolution, low resolution (480x270), and super-resolved low resolution. Lower-resolution images were upscaled to high resolution using bicubic interpolation.

Fig. 5. Example 200x200 detail from Witcher 3 at high resolution (1920x1080), medium resolution (854x480), super-resolved medium resolution, low resolution (480x270), and super-resolved low resolution. Lower-resolution images were upscaled to high resolution using bicubic interpolation.

2) Game-specific CNNs: We also wanted to characterize how much training models for specific games impacted our super-resolution results. We ran a full cross comparison of game models and validation sets for both medium- and low-resolutions. Our medium-resolution results are in Figure 6, and our low-resolution results are in Figure 7. The tables show the percent improvement in VMAF for either a medium- or low-resolution validation set. Each colored entry shows the percent change in VMAF from applying the row model to the column validation set. Each column is conditionally formatted, such that the greenest entry corresponds to the model that performed the best on the validation set, whereas the reddest entry corresponds to the model that performed the worst on that validation set. The bottom row of each table (i.e., Max-Min) shows the difference between the best and worst VMAF improvement.

For both tables, in nearly all cases the models that performed the best on a particular validation set were trained on frames from the same game. The exceptions to this pattern are in the medium-resolution results, in which the Witcher 3 model outperformed the Fallout 4 model on the Fallout 4 validation set (17.3% improvement vs 16.2%), the Rust model on the Rust validation set (18.6% to 15.8%), and the Starcraft 2 model on the Starcraft 2 validation set (15.2% to 15.1%).

The results also show that, like super-resolution in general, game-specific training is more effective for low-resolution streams. As noted above, training a model on different frames from the same game as the validation frames always created the best visual quality for low-resolution frames. And while mismatched models often performed well, in several cases they did not. For example, the difference between the matched model and the worst model for GTAV was 25.1%. For Starcraft 2 the difference was 16.5%, and for Witcher 3 the difference

Training	Validation						
	CSGO	Dota 2	Fallout 4	GTAV	Rust	Starcraft 2	Witcher 3
CSGO	19.1%	19.1%	16.6%	14.1%	17.8%	15.1%	20.4%
Dota 2	17.0%	19.5%	16.4%	15.7%	17.1%	13.1%	18.6%
Fallout 4	14.6%	16.5%	16.2%	15.6%	16.2%	5.5%	13.6%
GTAV	17.4%	18.7%	16.7%	18.2%	18.2%	13.9%	19.6%
Rust	16.7%	18.1%	16.5%	16.2%	15.8%	11.8%	17.8%
Starcraft 2	15.9%	16.6%	15.2%	14.4%	15.8%	15.1%	17.8%
Witcher 3	18.0%	19.5%	17.3%	17.3%	18.6%	15.2%	20.5%
Max-Min	4.5%	3.0%	2.1%	4.0%	2.8%	9.8%	6.9%

Fig. 6. VMAF results from cross validation of training/testing and validation sets for 854x480 frames.

Training	Validation						
	CSGO	Dota 2	Fallout 4	GTAV	Rust	Starcraft 2	Witcher 3
CSGO	81.2%	72.0%	59.1%	57.3%	73.8%	65.3%	84.1%
Dota 2	74.9%	75.9%	57.9%	75.4%	73.7%	64.5%	81.1%
Fallout 4	77.9%	74.0%	61.2%	79.7%	76.1%	59.5%	79.3%
GTAV	79.9%	73.3%	61.6%	82.4%	76.8%	66.0%	84.5%
Rust	75.5%	71.8%	61.9%	79.0%	78.3%	56.3%	76.0%
Starcraft 2	75.9%	72.3%	58.8%	74.7%	74.0%	72.8%	82.9%
Witcher 3	78.8%	74.8%	59.0%	79.7%	78.0%	69.4%	88.4%
Max-Min	6.3%	4.0%	4.0%	25.1%	4.6%	16.5%	12.4%

Fig. 7. VMAF results from cross validation of training/testing and validation sets for 480x270 frames.

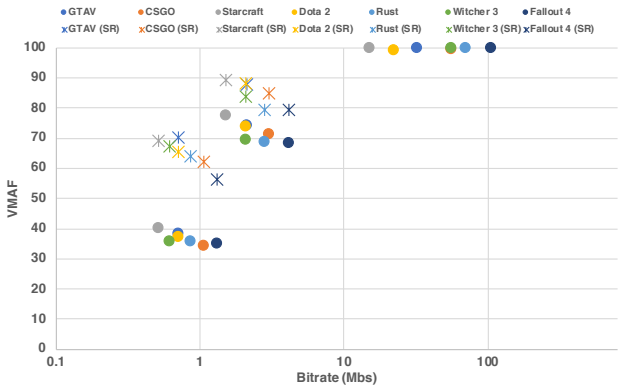

Fig. 8. Bitrate versus VMAF for game streams under H264 and super-resolved H264. Each dot represents an H264 stream, and each star represents a super-resolved stream. Dots and stars from the same game share the same color. The lowest-bitrate data points represent low-resolution videos (i.e., 480x270), the middle-bitrate data points represent medium-resolution videos (i.e., 854x480), and the highest bitrate data points represent high-resolution videos (i.e., 1920x1080). Note the logarithmic x-axis.

was 12.4%. The conclusion we draw from these results is that applying a game-specific model is almost always effective and provides insurance against using a poor-performing model.

These results are consistent with our hypothesis that LevelUp can benefit from training CNNs for the visuals of a particular game. At the same time, these results also suggest that a super-resolution CNN will improve visual quality even when given reduced-resolution images that are dissimilar from its training data.

3) Bandwidth-quality tradeoffs: In Section V-A1 we demonstrated that super-resolution CNNs can significantly improve the visual quality of reduced-resolution H264 videos. We would also like to understand the bandwidth and quality tradeoffs that LevelUp could offer. In particular, how do the visual quality and bitrate of a super-resolved video compare to a higher-resolution video?

Figure 8 shows the bitrate and VMAF for each game's H264 videos (the dots) and super-resolved videos (the stars). Note that since all super-resolution models are less than 250KB and could be downloaded in advance of streaming, we have not factored their size into the bitrates for super-resolved videos. Rather, super-resolution data points reflect the bitrates of the input H264 videos.

Our bitrate-VMAF results show that super-resolution cannot achieve the highest visual quality, and that the highest qualities require more than 10Mbs. Reducing videos' resolution leads to approximately 10x bandwidth savings for all streams. As expected, this bandwidth savings comes at the cost of reduced visual quality. In general, medium-resolution videos provide 10x bandwidth savings and 20-30% lower VMAF than high-resolution videos, whereas low-resolution videos provide 3x bandwidth savings and 50% lower VMAF compared to medium-resolution videos.

Though our results show that super-resolution CNNs cannot recover all of the visual quality lost from reducing a video's resolution, they show that these CNNs recover a great deal of it. In particular, for medium-resolution videos, rather than the 20-30% VMAF penalty of H264, super-resolved videos' quality is 10-20% of their high-resolution counterpart (at one-tenth the bitrate). The effect is more dramatic for low-resolution videos. Using low-resolution frames, super-resolution generates videos with VMAF values that are approximately 3-20% worse than medium-resolution H264, whereas raw low-resolution H264 provides visual quality that is 50% worse than medium-resolution H264.

Super resolution appears to be particularly effective for finely detailed game content, such as Witcher 3. As Figure 5 shows, super-resolution provides a much sharper image than simply interpolating upscaled low-resolution images. This sharpening effect has less impact on videos such as the Rust clip, which was dominated by a cloudless sky.

4) Discussion: Our results demonstrate that super resolution can improve the quality of reduced-resolution game streams. However, answers to the question of which video resolutions a LevelUp streamer should upload to the cloud depend on the streamer's bandwidth constraints. If the streamer lacks the bandwidth to upload at all resolutions or she needs to conserve bandwidth (e.g., due to data caps), then LevelUp could allow her to upload medium- and low-resolution streams without significantly compromising her audience's viewing experience. The same calculation applies to a LevelUp viewer. LevelUp's CNNs could allow viewers to use significantly less bandwidth to view a stream of slightly diminished quality.

B. Broadcaster performance

A key goal for LevelUp is to perform multi-bitrate video coding on a broadcaster's device instead of doing it in the cloud. To verify that multi-bitrate coding on a commodity device is feasible, we used our LevelUp prototype to livestream ten seconds of FortNite gameplay. For each test, we encoded and uploaded H264 videos at high resolution (1920x1080),

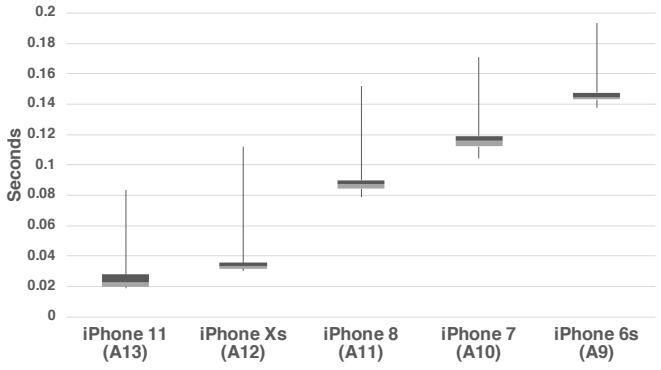

Fig. 9. Time to process frames while super-resolving low resolution (480x270) H264 screen recordings. The top edge of each dark-gray box represents the 75th percentile time, the bottom edge of each dark-gray box represents the median time, and the bottom edge of the light-gray box represents the 25th percentile time. The top whisker extends to the maximum time, and the bottom whisker extends to the minimum time.

medium resolution (854x480), and low resolution (480x270). LevelUp divided the ten seconds of gameplay into five two-second video segments, each encoded at 30FPS.

Apple's ReplayKit2 subsystem enforces strict resource limits on all broadcast extensions, including LevelUp's. In particular, LevelUp can use only 50MB of memory, so LevelUp synchronously processed each incoming video buffer to simplify memory management. After receiving a buffer, LevelUp placed it on a high-resolution encoding queue, a medium-resolution queue, and a low-resolution queue before returning. Once these queues filled to 60 buffers (i.e., two seconds at 30 FPS), LevelUp directed the encoder to save the three videos to disk. When each video was fully encoded, a completion handler for each video ran on a background thread and uploaded its video over a pre-initialized HTTP connection. For our experiments, we measured the time that LevelUp spent processing each screen buffer, starting from the time that the ReplayKit2 subsystem invoked LevelUp's buffer handler until the time that the handler returned. We performed our experiments on five generations of iPhones: the iPhone 11 Pro, the iPhone Xs, iPhone 8, iPhone 7, and iPhone 6s. All of these devices included an H264 hardware encoder.

As expected, all four devices processed livestream buffers in realtime, and the median processing time for all four devices was less than 1ms. Several devices had maximum processing times of close to 20ms due to the time to initialize the hardware encoder. Each of these outliers appeared only once and at the beginning of a run. Overall, our results demonstrate that modern mobile devices are capable of performing multi-bitrate video encoding in realtime.

C. Viewer performance

Multi-bitrate coding on mobile devices is not limited by devices' ability to encode videos in realtime, but platforms and broadcasters could be limited by bandwidth constraints. A broadcaster's connection may be too poor to upload high-, medium-, and low-resolution videos, or she may have wish to upload fewer bytes to remain under a data cap. A platform may wish to send less data to reduce the cloud-provider's transfer costs. Section V-A demonstrated that a super-resolution CNN can significantly improve the quality of reduced-resolution videos and could potentially mitigate bandwidth constraints, but for these CNNs to be practical, LevelUp viewers must be capable of super-resolving videos in realtime.

To determine whether LevelUp clients can super-resolve livestreams in realtime, we measured the time that our prototype spent processing frames of a Monument Valley livestream. For the experiment, we downloaded low-resolution video (i.e., 480x270) and processed each frame using Apple's CoreML subsystem. Prior to streaming, LevelUp pre-compiled and loaded a Monument Valley CNN for low-resolution images. To apply the model, incoming video frames had to be converted to grayscale. The CNN output a high-resolution, single-channel grayscale image (i.e., the Y luminance channel). LevelUp then merged this luminance channel with the chrominance channels (i.e., Cr and Cb) from the upscaled original frame. As in Section V-A, we upscaled the reduced-resolution Cr and Cb channels using bicubic interpolation.

Figure 9 shows our results. The iPhone 11 Pro processed frames with a median latency of 23ms and a 75-percentile of 28ms, The iPhone Xs processed frames with a median latency of 32ms and a 75th-percentile latency of 36ms. Both phones can display videos at approximately 30FPS. This demonstrates that a modern mobile device can super-resolve video in realtime. The 11 Pro and Xs's realtime performance are primarily due to their SoCs' Neural Engines. It is worth noting the significant performance improvement of the 11's A13 over the Xs's A12 (nearly 30% better median latency). As with our results in Section V-B, our viewer performance results show that mobile devices are ready to play a more central role in video streaming.

It is also worth noting that phones older than the iPhone Xs cannot deliver realtime CNN processing. The iPhone 8 has a two-core Neural Engine, and its median time to process frames was 87ms, with a 75th-percentile latency of 90ms. The iPhone 7 does not have a dedicated neural-net accelerator and executes the LevelUp CNN on its six-core GPU.

D. Energy overhead

Precisely measuring the energy consumption of a modern device like an iPhone is challenging. Like the vast majority of modern smartphones, iPhones' batteries are not easily accessible, which prevented us from attaching a power monitor such as a Monsoon. iOS provides coarse-grained energy and performance monitoring, but these reports only measure energy intensity on a scale from 1 to 10.

Despite these limitations, we ran a series of simple experiments on an iPhone Xs and iPhone 11 Pro with the game Monument Valley. First, we used LevelUp to record five minutes of Monument Valley game play. Then we viewed this stream under several configurations for 30 minutes with all radios on and the screen at 25% intensity. When we reached the end of the five-minute stream, we looped to

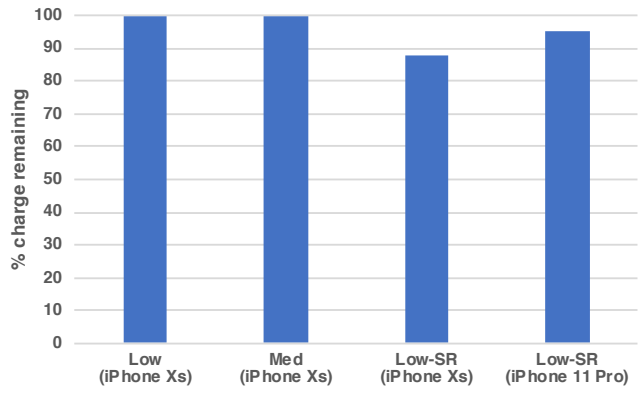

Fig. 10. Percent battery remaining after streaming 30 minutes of Monument Valley gameplay on an iPhone Xs and iPhone 11 Pro.

continue streaming from the beginning. Thus, each streaming experiment played our Monument Valley recording six times. Before each streaming session, we charged the phone battery to 100%. We then unplugged the device, streamed video for 30 minutes from a LevelUp server over WiFi, and finally recorded how much battery charge remained. For these experiments, we streamed low-resolution video and applied super-resolution (low-sr), low-resolution video without super-resolution (low), and medium-resolution video without super-resolution (med). Figure 10 shows our results.

Unsurprisingly, applying super resolution required more energy than simple streaming, leaving 87% of the battery after 30 minutes on the iPhone Xs and 95% left on the iPhone 11 Pro. This was due to the additional computational burden of super-resolving low-resolution frames and merging the results with the original frames. It is also unsurprising that streaming low-resolution videos is energy efficient since these videos require less compute to decode and display and less bandwidth to download: streaming low-resolution video left 100% of the phone's battery, and streaming medium-resolution video left 100% of the phone's battery. The percent of remaining battery remaining on the 11 Pro compared to the Xs is noteworthy. We suspect that this is due to the 11 Pro's larger battery as well as a more power efficient ML-accelerator.

Our results suggest that for the most recent devices (e.g., the iPhone 11 Pro), super resolution will have a very small impact on users' experience. However, for older devices like the Xs, when energy is scarce, the additional boost in quality offered by super resolution may be offset by noticeable energy loss. Thus, using super resolution makes the most sense when battery power is plentiful (e.g., a device is fully charged) and network bandwidth is constrained or precious (e.g., over a weak cellular connection). However, assuming that a device's battery drains linearly, our results also show that someone who begins watching a stream with a fully charged, modern device could watch five hours of video and have close to half of her battery charge left.

VI. Related work

There has been an great deal of recent interest in improving video streaming. Much of this work has focused on improving the systems that deliver video streams instead of only focusing on greater coding efficiency. Specifically, architectures are increasingly exploring ways to exploit resources on both the client and server sides.

For example, NAS [24] uses a DNN to learn the mappings from low quality to high quality videos, including super-resolution. The client downloads and uses the DNN to transform lower quality images into higher quality ones. Multiple scalable DNNs are used to meet the resource requirements of heterogeneous client environments. One of the challenges that NAS faces that LevelUp does not is how to handle very large models. NAS model are close to 100MB, whereas LevelUp models are only 250KB. This has two implications. First, NAS models are too large for the ML co-processor of a smartphone, and there can only run on a machine with a desktop-class GPU. Second, NAS includes complex logic for balancing the bandwidth spent downloading video content and the bandwidth downloading models. LevelUp models are so small that they easily execute on a smartphone and do not require complex logic to download.

CloudSeg [20] and LiveNAS [13] both try to overcome broadcasters' bandwidth limitations by boosting the quality of low-bitrate video with super resolution. Of the two, LiveNAS is more relevant to LevelUp, although both use server-side super-resolution to create higher bitrate versions of live video. LiveNAS enlists clients' help in maintaining the super-resolution models; broadcasters send samples of their local high-quality content to the server to help update the super-resolution model at broadcast time. Many of the techniques proposed by LiveNAS could be integrated into LevelUp. In particular, enlisting clients to provide super-resolution training data could help LevelUp create and maintain its models. LevelUp most differs from CloudSeg and LiveNAS in who performs super-resolution and who performs multi-bitrate transcoding. CloudSeg and LiveNAS rely on server-side infrastructure for these tasks, whereas LevelUp investigates the feasibility of doing this work on the mobile edge.

Dejavu [12] leverages similar insights and techniques as LevelUp to enhance video conferencing. The primary difference between LevelUp and Dejavu is the target application. Whereas a video-conferencing system focuses on the visual features of video chat (e.g., participants' faces), LevelUp focuses on the visual features of game content (e.g., game characters and settings).

Like LevelUp, Kahawai [6] focuses on gaming and saving bandwidth. However, Kahawai is a cloud gaming system that offloads part of the mobile GPU's workload to the server. The server creates and sends a "patch" video to the client. The client then applies the patch, either "delta" frames, or I-frames, to the local images to improve game visual quality. It is possible that the technique used in LevelUp could also be applied to cloud gaming, but we leave this for future work.

Choosing the right video streaming bitrate under different network conditions is a well known research topic. This is a hard problem because a chosen bitrate must balance two seemly conflicting goals: maximizing quality and minimizing re-buffering time. Pensieve [17] uses reinforcement learning to learn a control policy for adaptively choosing streaming bitrates. A neural network expresses different system observations in the control policy so that it can adapt to networks of different characteristics automatically. LevelUp is orthogonal to this and other work on policies for stream adaptation.

Other recent work introduces novel codec/network interfaces for video processing platforms. Salsify [9] is a video conferencing system that uses a pure functional codec so that the encoder state can "fork" into different execution branches and delay the choice of frame quality as late as possible. This allows it to swiftly and precisely adapt to network condition changes. However, it is only designed for one-to-one video conferencing, not one-to-many video streaming like LevelUp.

ExCamera [10] uses a similar functional video codec, but instead of exploring multiple execution branches, it exposes the internal state of each thread so that videos can be encoded with more parallelism using serverless platforms without losing compressing efficiency. Sprocket [5] extends ExCamera so that it can not only encode videos but also allow users to build more complex video process pipelines, including those with machine learning applications like facial recognition. These pipelines are then orchestrated by Sprocket to run on serverless platforms with high parallelism and low cost. LevelUp occupies a decidedly different point in the design space than either ExCamera or Sprocket since it relies on mobile devices' hardware encoders to perform multi-bitrate transcoding.

VII. Conclusion

We have presented the design and implementation of LevelUp. LevelUp embodies a thin-cloud approach to game livestreaming. LevelUp broadcasters use hardware accelerators to generate videos at multiple levels of quality and upscale reduced-resolution streams. Experiments with a prototype implementation demonstrate that mobile hardware acceleration can transcode and super-resolve video in realtime, and that super resolution can improve the visual quality of low-resolution game streams by up to 88%.

References

[1] Toward a practical perceptual video quality metric. http://techblog.netflix.com/2016/06/toward-practical-perceptual-video.html, 2016.

[2] Alliance for open media. https://aomedia.org, 2018.

[3] Performance of samsung and apple flagship smartphones: 43.5 million active iphone x devices in may 2018. https://newzoo.com/insights/articles/performance-of-samsung-and-apple-flagship-smartphones/, 2018.

[4] iphone 11 surpasses iphone xr to become worlds most-popular smartphone model in q1. https://www.omdia.com/resources/product-content/iphone-11-surpasses-iphone-xr-to-become-worlds-most-popular-smartphone-model-in-q1, 2020.

[5] L. Ao, L. Izhikevich, G. M. Voelker, and G. Porter. Sprocket: A serverless video processing framework. In *Proceedings of the ACM Symposium on Cloud Computing*, pages 263–274. ACM, 2018.

[6] E. Cuervo, A. Wolman, L. P. Cox, K. Lebeck, A. Razeen, S. Saroiu, and M. Musuvathi. Kahawai: High-quality mobile gaming using gpu offload. In *Proceedings of the 13th Annual International Conference on Mobile Systems, Applications, and Services*, pages 121–135. ACM, 2015.

[7] C. Dong, C. C. Loy, K. He, and X. Tang. Image super-resolution using deep convolutional networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(2):295–307, Feb 2016.

[8] W. Dong, L. Zhang, G. Shi, and X. Wu. Image deblurring and super-resolution by adaptive sparse domain selection and adaptive regularization. *Transactions of Image Processing*, 20(7):1838–1857, July 2011.

[9] S. Fouladi, J. Emmons, E. Orbay, C. Wu, R. S. Wahby, and K. Winstein. Salsify: Low-latency network video through tighter integration between a video codec and a transport protocol. In *15th Symposium on Networked Systems Design and Implementation*, 2018.

[10] S. Fouladi, R. S. Wahby, B. Shacklett, K. Balasubramaniam, W. Zeng, R. Bhalerao, A. Sivaraman, G. Porter, and K. Winstein. Encoding, fast and slow: Low-latency video processing using thousands of tiny threads. In *NSDI*, pages 363–376, 2017.

[11] D. Glasner, S. Bagon, and M. Irani. Super-resolution from a single image. In *ICCV*, 2009.

[12] P. Hu, R. Misra, and S. Katti. Dejavu: Enhancing videoconferencing with prior knowledge. In *Proceedings of the 20th International Workshop on Mobile Computing Systems and Applications*, HotMobile '19, pages 63–68, New York, NY, USA, 2019. ACM.

[13] J. Kim, Y. Jung, H. Yeo, J. Ye, and D. Han. Neural-enhanced live streaming: Improving live video ingest via online learning. In *Proceedings of the Annual Conference of the ACM Special Interest Group on Data Communication on the Applications, Technologies, Architectures, and Protocols for Computer Communication*, SIGCOMM '20, page 107125, New York, NY, USA, 2020. Association for Computing Machinery.

[14] J. Kim, J. K. Lee, and K. M. Lee. Accurate image super-resolution using very deep convolutional networks. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1646–1654, 2016.

[15] S. Li, F. Zhang, L. Ma, and K. N. Ngan. Image quality assessment by separately evaluating detail losses and additive impairments. *IEEE Transactions on Multimedia*, 13(5):935–949, 2011.

[16] B. Lim, S. Son, H. Kim, S. Nah, and K. M. Lee. Enhanced deep residual networks for single image super-resolution. In *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops, CVPR Workshops 2017, Honolulu, HI, USA, July 21-26, 2017*, pages 1132–1140, 2017.

[17] H. Mao, R. Netravali, and M. Alizadeh. Neural adaptive video streaming with pensieve. In *Proceedings of the Conference of the ACM Special Interest Group on Data Communication*, pages 197–210. ACM, 2017.

[18] H. R. Sheikh and A. C. Bovik. Image information and visual quality. In *Acoustics, Speech, and Signal Processing, 2004. Proceedings.(ICASSP'04). IEEE International Conference on*, volume 3, pages iii–709. IEEE, 2004.

[19] W. Shi, J. Caballero, F. Huszar, J. Totz, A. P. Aitken, R. Bishop, D. Rueckert, and Z. Wang. Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pages 1874–1883, 2016.

[20] Y. Wang, W. Wang, J. Zhang, J. Jiang, and K. Chen. Bridging the edge-cloud barrier for real-time advanced vision analytics. In *11th USENIX Workshop on Hot Topics in Cloud Computing (HotCloud 19)*, Renton, WA, July 2019. USENIX Association.

[21] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612, 2004.

[22] C.-Y. Yang, C. Ma, and M.-H. Yang. Single-image super-resolution: A benchmark. In *Proceedings of European Conference on Computer Vision*, 2014.

[23] J. Yang, J. Wright, T. S. Huang, and Y. Ma. Image super-resolution via sparse representation. *IEEE Transactions on Image Processing*, 19(11):2861–2873, Nov 2010.

[24] H. Yeo, Y. Jung, J. Kim, J. Shin, and D. Han. Neural adaptive content-aware internet video delivery. In *13th Symposium on Operating Systems Design and Implementation*, pages 645–661, 2018.