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 real-time 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.

1. INTRODUCTION

Game streaming services are large and growing. As of February 2018, Twitch reports having over 140 million unique monthly viewers.1 And between 2017 to 2019 Twitch increased the average number of concurrent viewers from nearly 750,000 to over 1.2 million. It also doubled the average number of concurrent streams from 25,000 to 50,0002. Yet despite the popularity of Twitch and other livestreaming platforms like Facebook Live, livestreaming is costly.

Transcoding 100 hours of multi-bitrate live video on AWS and Azure costs between $300 and $500, and transferring video data to viewers can cost thousands of dollars per channel, per month. These costs are prohibitive for small companies looking to grow new platforms, and represent a significant savings opportunity for large companies like Amazon, Facebook, and Microsoft.

One reason for these costs is the expense of performing real-time, multi-bitrate transcoding on cloud infrastructure. Wowza charges less than $20 to livestream 100 hours of single-bitrate video, which is roughly 15x cheaper than 100 hours of multi-bitrate video. Multi-bitrate services typically accept video streams from a broadcaster at a single bitrate and then transcode the video at multiple bitrates so that viewers can adapt their stream quality to network conditions. When a viewer's network is strong, it will stream high-quality video, and when a viewer's network is weaker, it will shift to lower-quality video. Generating multiple video qualities in real-time is computationally intensive and requires specialized hardware like graphical processing units (GPUs) or dedicated hardware transcoders to keep streaming latency low. Unlike commodity cloud services like web front-ends, blob storage, and content distribution networks (CDNs), real-time video transcoding has not achieved economies of scale to drive down costs. In addition, transfer costs are high due to platforms' large audiences and the amount of time these audiences spend watching streams. For example, the total number of minutes viewed on Twitch grew from 355 billion in 2017 to over 600 billion in 20193.

In this paper, we argue that edge devices are ready to play a more central role in livestreaming platforms. We propose a new approach to livestreaming based on the notion of a thin cloud, in which the cloud provides commodity storage and content-distribution infrastructure, but is not relied upon to provide expensive resources like 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

---

1twitchadvertising.tv
2twitchtracker.com/statistics
3twitchtracker.com/statistics
responsible for storing and distributing video and other
files, and edge devices assume responsibility for multi-
bitrate transcoding and boosting the quality of reduced
bitrate streams.

Our approach is enabled by the rapid deployment of
hardware acceleration on the edge. Because gaming,
photography, and media playback are critical smart-
phone applications, GPUs and video-transcoding hard-
ware is commonplace on smartphones SoCs. Further-
more, the next wave of accelerators, machine learn-
ing (ML) co-processors, have already been deployed on
high-end smartphones. These accelerators can handle
complex ML workloads and will be pervasive in the near
future.

LevelUp uses 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 stream viewers can 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 a gamin-
g platform must transfer to clients without sacrificing
visual quality.

We do not claim any contributions to the field of
deep learning or ML. Work on compressing models to
run more effectively on mobile devices and construct-
ing models that achieve better inference and/or perfor-
mance are orthogonal to LevelUp. Instead, our contri-

• We identify multi-bitrate transcoding and data trans-
fer as costly parts of cloud-based livestreaming,
and propose a thin-cloud approach to streaming
that pairs commodity web infrastructure in the
cloud and commodity hardware-acceleration on the
edge.

• We identify super-resolution as a way to mitigate
the high cost of transferring game-stream data. In
particular, we observe that super-resolution mod-
els trained with specific game content can pro-
vide additional quality improvements over networks
trained on different games.

• Using our LevelUp prototype, we show that real-
time multi-bitrate transcoding is feasible on com-
modity devices, and that high-end SoCs can per-
form realtime super-resolution of reduced-resolution
videos. We further demonstrate that super-resolution

The rest of the paper is organized as follows: Sec-
tion 2 provides background information on livestream-
ing, mobile-device hardware, and image-similarity met-
rics; Section 3 articulates the design principles under-
lying LevelUp, Section 4 describes the LevelUp design,
Section 5 describes our experimental evaluation, Sec-
tion 6 describes related work, and Section 7 provides
our conclusions.

2. BACKGROUND

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

2.1 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 their 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 also
maintains a persistent TCP connection to the RTMP
server, and the RTMP server pushes new chunks to its
viewers as they become available. In its simplest form,
an RTMP server acts as a pass-through relay for me-
dia chunks from a broadcaster to a viewer. RTMP also
supports multi-bitrate coding, in which an RTMP server
transcodes incoming media into different qualities (e.g.,
different resolutions) before forwarding the appropriate
chunks to viewers. Viewers are responsible for commu-
icating their desired level of quality to the server.

The primary benefit of RTMP is that it can pro-
vide low end-to-end latency between a broadcaster and
player. There are at least two reasons for this. First, be-
cause RTMP pushes media over a persistent TCP con-
nection, servers can forward chunks as soon as they are
available. This is much faster than the alternative of
forcing viewers to poll for new chunks, or worse, initiat-
ing a new connection for each poll. Second, each small
chunk in an RTMP stream necessarily covers a brief pe-
riod of time. 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 broad-
caster can send new data to the RTMP server every
128ms.

Despite its low latency, RTMP has a major draw-
back. Because RTMP runs directly on top of TCP it
requires a dedicated server, and thus RTMP does not
easily integrate with existing content distribution net-
works (CDNs). In contrast, HLS and Dynamic Adap-
tive Streaming over HTTP (MPEG-DASH) scale more

Like RTMP, HLS divides video streams into smaller
segments. However, because HLS runs on top of HTTP
rather than directly on TCP, segments are stored as normal files (e.g., an H264 video) and viewers must retrieve new segments by issuing HTTP get requests. HTTP/2 supports server push, but HTTP/2 is not widely deployed or integrated with HLS, and thus viewers must pull video segments over HTTP.

HLS segments must be relatively large compared to RTMP chunks in order to amortize the cost of issuing HTTP requests. HLS segments typically capture 2-10 seconds of content, and these longer segment lengths guarantee 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 with large followings who stream on Facebook Live and popular gamers who stream on Twitch will attract a large enough audience (e.g., hundreds if not thousands or millions of viewers) that interactivity on the order of milliseconds is simply not possible. While lower latency is always better, the low delays offered by RTMP are often overkill for streams that scale beyond tens of participants.

In order for a viewer to retrieve the next segment over HTTP, it must know the segment’s URL. 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 can then adapt the quality of their stream by downloading the segments that best match their network conditions.

Importantly, HLS scales well because playlist and video-segment files can be distributed through existing web CDNs. MPEG-DASH is similar to HLS and offers the same easy integration with commodity web infrastructure. This is a major reason why 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 so it runs entirely over HTTP like HLS and MPEG-DASH. However, while HLS can leverage some commodity web infrastructure, one feature of livestreaming still requires (relatively) special-purpose machinery: multi-bitrate streaming. For both RTMP and HLS, broadcasters upload video at the best bitrate they can, and the server is responsible for generating different video qualities so that viewers can adapt their stream.

Transcoding live video in the cloud is expensive. For example, 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 data transfer costs). 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. A primary reason that multi-bitrate transcoding in the cloud is so expensive is that transcoding in real time requires hardware acceleration, such as expensive GPUs or other special-purpose hardware. In addition, the cost to deliver high-quality videos to clients at even moderate scale can also be expensive. Azure charges between $7,500 and $5,000 to transfer 150-500 TB/month. AWS Media Connect prices are nearly identical.

We believe that mobile devices can have an important role to play in driving down these costs. LevelUp explores this idea by fully utilizing commodity smartphone hardware to (1) eliminate cloud-based transcoding, and (2) significantly improve the visual quality of lower bitrate videos.

### 2.2 Mobile hardware accelerators

System-on-chip (SoC) and smartphone manufacturers deploy hardware accelerators to meet the evolving computational demands of mobile applications. As it became clear that gaming was an important application for smartphones, devices of all qualities began to ship with powerful GPUs. Similarly, as smartphones became users’ primary device for taking photos and videos, SoCs integrated dedicated hardware for video encoding and decoding, as well as advanced image processing. 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 far more widely deployed on smartphones than it is 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. Thus, just among active iPhones there are over 300 million processors capable of hardware-accelerated video transcoding.

---

4 As of 12-2019: azure.microsoft.com/en-us/pricing/details/media-services/
5 As of 12-2019: aws.amazon.com/medialive/pricing/
6 newzoo.com/insights/articles/performance-of-samsung-and-apple-flagship-smartphones/
ing. While individual computational units in the cloud may be more capable than their mobile counterparts due to power constraints (e.g., GPUs), the cloud will never reach the same scale of deployment that can be achieved in aggregate 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 then it will be more cost effective to perform this work on commodity mobile devices than on dedicated cloud infrastructure.

2.3 Image-quality metrics

To properly evaluate whether LevelUp can significantly boost the quality of low-bitrate videos, one must have a way to measure image quality. 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 its original representation. Image quality is a subjective matter, and image-quality metrics are a way 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 information than to loss of chrominance information.

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 well known limitations, and researchers proposed the Structural Similarity Index (SSIM) as an alternative to PSNR nearly 15 years ago [17]. SSIM is more complex than PSNR and tries to capture the perceived change in structural information caused by an image transformation. Though SSIM is generally seen as an improvement over PSNR, both fail to consistently reflect human perception [1]. Of particular concern for LevelUp, measuring the quality of individual frames may not capture the quality of an entire video.

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) [15], Detail Loss Metric (DLM) [12], 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. In doing so, VMAF tries to avoid the weaknesses of individual metrics and approximate how a human would holistically evaluate a video’s quality. We primarily rely on VMAF to evaluate LevelUp.

3. DESIGN PRINCIPLES

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

Transcode videos on broadcast devices. As discussed in Section 2.1, existing cloud services charge an order of magnitude more for multi-bitrate livestreaming than for single-bitrate livestreaming. And as discussed in Section 2.2, nearly all mobile devices possess dedicated hardware for encoding and decoding video. Thus, LevelUp clients 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. Games require a device’s screen, CPU cores, and GPU to all 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 medium- and low-resolution videos is small compared to high-resolution videos. For example, our results in Section 5.1.3 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 it may be due to more persistent issues like a network data cap. Similarly, platforms may wish to limit their egress bandwidth by serving lower-bitrate streams. LevelUp mitigates the effect of such constraints through
Mitigate lower bitrates 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 throttle their load 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 5.1.3, uploading medium- and low-resolution videos would save significant bandwidth compared to uploading all three qualities. Transferring lower bitrate video to viewers can also be a large source of saving for platforms. Of course, reduced-resolution videos look worse than high-resolution videos. But like LevelUp broadcasters, LevelUp viewers also have sophisticated hardware accelerators, and they can use these accelerators to mitigate the reduced visual quality of low-bitrate video.

Single-image super-resolution is an ML approach to improving the quality of reduced-resolution images [9, 18]. Unlike a simple interpolation, super-resolution methods aim 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 [6, 19], and more recent work has investigated training neural networks to perform super resolution [5, 11, 13, 16].

LevelUp uses the convolutional neural network (CNN) for super resolution described in [16]. 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 only higher-end smartphones like the latest iPhones and Google Pixels, but this hardware will trickle down to all device levels within a few years.

LevelUp is not the first system to use super-resolution for video streaming. NAS [20] 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 [10] leverages similar insights and techniques to enhance video conferencing. However, 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. We use to train game CNNs with streams from prior sessions.

These CNNs learn general rules for upsampling 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? In this paper we seek only to validate our hypothesis that super-resolution can be applied to game content and that there is additional benefit to training for a specific game. We leave the many remaining questions for future work.

4. DESIGN

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.

4.1 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.
Figure 1: High-level LevelUp architecture. A broadcaster device uses its local hardware video encoders to generate high-, medium-, and low-resolution videos. It uploads these files to cloud storage, where they can be distributed to viewers via commodity web CDN. Viewers use a stream playlist to identify the locations of each resolution stream, as well as the appropriate super-resolution model to apply the stream it is viewing.

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 queues 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.

4.2 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. 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 [14] 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.

4.3 Server

The final component of the LevelUp architecture is the server. By design, the LevelUp server is extremely simple. Our current implementation is a simple 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.

5. 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.org⁷: Counter-Strike, Global Offensive (CSGO); Dota II; 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 STARCAST2) 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 [16].

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 either viewed a stream of or played Monument Valley, beginning with a full battery, and recorded the remaining battery capacity after 30 minutes.

All videos in our experiments were encoded using H264. We did this 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 encoding algorithm with hardware-accelerated decoding.

5.1 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 ll testing, training, and validation frames using ffmpeg at either medium resolution (i.e., 854x480) or 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 an 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 a
We fed each game’s CNN decoded frames 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 videos using a model trained and validated on the same game (though never the same frames).

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 2.3 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.

### 5.1.2 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 score for either a medium- or low-resolution validation set. Each colored entry shows the percent change in VMAF from apply-
Table 1: VMAF results from cross validation of training/testing and validation sets for 480x270 frames.

<table>
<thead>
<tr>
<th>Training</th>
<th>CSGO</th>
<th>Dota 2</th>
<th>Fallout 4</th>
<th>GTAV</th>
<th>Rust</th>
<th>Starcraft 2</th>
<th>Witcher 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>CSGO</td>
<td>19.1%</td>
<td>19.1%</td>
<td>16.6%</td>
<td>14.1%</td>
<td>17.8%</td>
<td>15.1%</td>
<td>20.2%</td>
</tr>
<tr>
<td>Dota 2</td>
<td>17.0%</td>
<td>19.5%</td>
<td>16.4%</td>
<td>15.7%</td>
<td>17.1%</td>
<td>13.1%</td>
<td>18.6%</td>
</tr>
<tr>
<td>Fallout 4</td>
<td>14.8%</td>
<td>18.5%</td>
<td>16.2%</td>
<td>15.6%</td>
<td>16.7%</td>
<td>13.0%</td>
<td>13.0%</td>
</tr>
<tr>
<td>GTAV</td>
<td>17.4%</td>
<td>18.7%</td>
<td>16.7%</td>
<td>18.4%</td>
<td>18.2%</td>
<td>13.9%</td>
<td>19.6%</td>
</tr>
<tr>
<td>Rust</td>
<td>16.1%</td>
<td>18.1%</td>
<td>16.5%</td>
<td>16.2%</td>
<td>15.8%</td>
<td>11.8%</td>
<td>17.8%</td>
</tr>
<tr>
<td>Starcraft 2</td>
<td>15.9%</td>
<td>16.8%</td>
<td>15.2%</td>
<td>14.8%</td>
<td>15.8%</td>
<td>15.8%</td>
<td>17.0%</td>
</tr>
<tr>
<td>Witcher 3</td>
<td>18.6%</td>
<td>19.5%</td>
<td>17.9%</td>
<td>17.5%</td>
<td>18.6%</td>
<td>15.2%</td>
<td>20.5%</td>
</tr>
</tbody>
</table>

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

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 upsampled low-resolution images. This sharpening effect has less impact on games such as Rust clip, which was dominated by a cloudless sky.

5.1.4 Discussion

Our results demonstrate that super resolution can im-
prove 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.

5.2 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), 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, with 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.

5.3 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

---

Figure 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.

Figure 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.
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 5.1 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 upsampled original frame. As in Section 5.1, we upsampled 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 5.2, our viewer performance results show that mobile devices are ready to play a more central role in video streaming.

It is 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 75-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.

5.4 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 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 latest 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 in settings where 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.

6. 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. This approach has proved effective in several recent systems, including LevelUp.

For example, NAS [20] 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.

Dejavu [10] 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 videogames (e.g., game characters and settings).

Like LevelUp, Kahawai [4] 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 seemingly conflicting goals: maximizing quality and minimizing re-buffering time. Pensieve [14] 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 [7] 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 [8] 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 [3] extends ExCamera so that it can not only encode videos but also allow users to build more complex video processing 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.

7. 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 real-time, and that super resolution can improve the visual quality of low-resolution game streams by up to 88%.

8. REFERENCES


