Neural Adaptive Content-aware Internet Video Delivery

Hyunho Yeo, Youngmok Jung, Jaehong Kim, Jinwoo Shin, and Dongsu Han, *KAIST*

https://www.usenix.org/conference/osdi18/presentation/yeo

This paper is included in the Proceedings of the 13th USENIX Symposium on Operating Systems Design and Implementation (OSDI '18).

October 8–10, 2018 • Carlsbad, CA, USA

ISBN 978-1-939133-08-3

Open access to the Proceedings of the 13th USENIX Symposium on Operating Systems Design and Implementation is sponsored by USENIX.

Neural Adaptive Content-aware Internet Video Delivery

Hyunho Yeo Youngmok Jung Jaehong Kim Jinwoo Shin Dongsu Han
KAIST

Abstract

Internet video streaming has experienced tremendous growth over the last few decades. However, the quality of existing video delivery critically depends on the bandwidth resource. Consequently, user quality of experience (QoE) suffers inevitably when network conditions become unfavorable. We present a new video delivery framework that utilizes client computation and recent advances in deep neural networks (DNNs) to reduce the dependency for delivering high-quality video. The use of DNNs enables us to enhance the video quality independent to the available bandwidth. We design a practical system that addresses several challenges, such as client heterogeneity, interaction with bitrate adaptation, and DNN transfer, in enabling the idea. Our evaluation using 3G and broadband network traces shows the proposed system outperforms the current state of the art, enhancing the average QoE by 43.08% using the same bandwidth budget or saving 17.13% of bandwidth while providing the same user QoE.

1 Introduction

Internet video has experienced tremendous growth over the last few decades. Recent market reports indicate people around the world watch 5.75 hours of online video per week on average [10] and video traffic is expected to quadruple in the next five years [26, 63]. Current video delivery infrastructure has been successful in handling the scalability challenges with two key technologies. First, at the server side, distributed computing technologies enabled content delivery at Internet scale. Second, at the client side, adaptive bitrate (ABR) streaming addressed the problem of bandwidth heterogeneity and its variations across time and space. Techniques at both ends evolved over time to optimize user quality of experience (QoE) as it ultimately impacts the revenue of various stakeholders [22, 27, 77].

However, the limitation of existing content distribution networks (CDNs) is that its quality heavily depends on the bandwidth between servers and clients. When the bandwidth resource becomes scarce, user QoE suffers directly [43, 47]. Bitrate adaptation has been the primary tool to relax the problem [52]. Nevertheless, its sole reliance on network resource is a fundamental limitation.

Inspired by the ever-increasing clients' computational power and recent advances in deep learning, this paper identifies an alternative and complementary approach to enhancing the video quality. We apply a deep neural network (DNN)-based quality enhancement on video *content* utilizing the *client computation* to maximize user QoE. In particular, a deep learning model learns a mapping from a low-quality video to a high-quality version, e.g., super-resolution. This enables clients to obtain high-definition (e.g., 1080p) video from lower quality transmissions, providing a powerful mechanism for QoE maximization on top of bitrate adaption.

Leveraging client computation via DNNs impacts the server/client system and introduces a number of non-trivial challenges:

- First, the CDN servers have to provide a DNN model for the content they provide. However, it is difficult to guarantee the test performance of DNN's predictions. It is especially unreliable for unseen/new content, presenting a significant barrier to deployment.
- Second, client devices are heterogeneous. Their computational power varies widely and may even exhibit temporal variation due to multiplexing. Nevertheless, DNN-based quality enhancement must occur at real-time to support online video streaming.
- Finally, the DNN-based quality enhancement has a cascading effect on ABR-based QoE optimization. The quality now depends on the availability of DNNs at the client in addition to the available bandwidth. Thus,

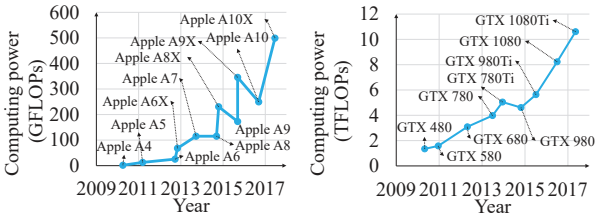
existing ABR algorithms must reflect the changes.

This paper presents NAS, the first video delivery framework that applies DNNs on video content using client's computational power to maximize user QoE. We present a system design that runs on top of Dynamic Adaptive Streaming over HTTP (DASH) framework. NAS addresses the challenges by introducing new system designs. To guarantee reliable quality enhancement powered by DNN, it takes a content-aware approach in which a DNN is trained for each content separately. The idea is to leverage the DNN's overfitting property and use the training accuracy to deliver predictable high performance, instead of relying on the unpredictable test accuracy. Next, to meet the real-time constraints on heterogeneous environments, we use multiple scalable DNNs that provide anytime prediction [24, 36]. Such DNN architectures can adaptively control their computational cost given resource budget. NAS clients choose a DNN (from multiple options) that best fits their resources and adapt to temporal variations in computing power at each time epoch. The scalable DNN also enables the use of a partially downloaded DNN, bringing an incremental benefit in downloading a DNN model. Finally, to reconcile the ABR-based QoE optimization and DNN-based quality enhancement, we devise a content enhancement-aware ABR algorithm for QoE optimization. To this end, we integrate our design into the state-of-the-art ABR algorithm [52] that uses reinforcement learning [68]. The algorithm decides when to download a DNN model and which video bitrate to use for each video chunk.

We evaluate NAS using a full system implementation. Our evaluation on 27 real videos and 17.8 hours of real-world network traces [8] using six different GPU models shows NAS delivers substantial benefit in a wide range of settings and is able to meet the real-time constraint on desktop class GPUs of varying capacity. In particular, it improves user QoE between 63.80-136.58% compared to BOLA [66] used in DASH [4] and between 21.89-76.04% compared to Pensieve, the state-of-the-art ABR design. Finally, we provide in-depth performance analysis of individual system components.

In summary, we make three key contributions:

- **End-to-end video delivery system:** NAS is an end-to-end video streaming system that integrates the content-aware approach, DNNs for super-resolution, scalable anytime prediction, and mechanisms for handling device heterogeneity on top of an existing adaptive streaming framework.
- **Use of DNNs in adaptive streaming:** NAS is the first system to apply super-resolution DNNs over video content in the context of adaptive streaming. From the machine learning (ML) side, we are the first to apply DNN-streaming, super-resolution, and anytime prediction to adaptive streaming.
- **Content-aware DNN:** NAS streams video along with the corresponding content-aware DNN to its clients. This is a key enabler and a novel component of NAS, which can be also viewed as a new approach to video coding.



(a) Mobile (b) Desktop

Figure 1: Growth of GPU's processing power

2 Motivation and Goal

Traditional approaches to improving video stream quality include: using better codecs [11, 12]; optimizing adaptive bitrate algorithms [20, 39, 42]; choosing better servers and CDNs [17, 50, 74]; and using coordination among clients and servers through a centralized control plane [51, 54]. These approaches focus on how to best utilize the network resource, but suffer from two common limitations.

Under-utilization of client's computation. Market reports [10, 57] indicate the majority of users watch video primarily on PCs, which have significant computation power. Mobile devices, which is the next popular platform, are also equipped with power-efficient graphic processing units (GPUs) [29]. Figure 1 shows the exponential growth in GPU's computing power over time on mobile devices and desktop PCs. Latest mobile devices even have dedicated hardware for neural processing [7]. However, the current video delivery infrastructure *underutilizes* client's computational power. With their growing computational capacity and ever-increasing demand for bandwidth, we envision a video delivery system in which clients take an active role in improving the video quality.

Limitation of current video coding. Video episodes often contain redundancy that occurs at large timescales. For example, consider a popular sports game (e.g., NBA finals) watched by millions of people. Same objects (e.g., balls and players) and scenes (e.g., basketball court) show up repeatedly. Similarly, redundancy is also found within episodes of a TV show, games in a sports league, and videos from the same streamers. Such frequently reoccurring high-level features contain valuable information that can be leveraged for video coding. However, standard

video coding, such as MPEG and H.26x, only captures spacial and short-term redundancy, lacking any mechanisms to exploit motion picture's high-level features. Within a group of pictures (GOP), inter-frame coding encodes the difference between adjacent frames to compress a motion picture [30]. However, a GOP is typically on the order of seconds for online video [13], making it impossible to capture redundancy that occurs at large timescales. As long as codecs compress video only within a GOP (arguably a fundamental constraint for streaming), using sophisticated codecs would not completely close this gap.

Motivated by this, we envision a video delivery system that exploits such redundancy by capturing the high-level features and applies additional client computation to augment the limitation of traditional video encoding. To this end, we utilize DNNs that abstract meaningful features from a low-level representation of data [23].

System goal. Our goal is to design a practical system that augments the existing infrastructure to optimize user QoE. As the first step, we consider servicing on-demand videos, as opposed to live streams, and using personal computers that have desktop-class GPUs. We propose a redesign of the video delivery infrastructure to take advantage of client computation to a greater degree. For quality enhancement, we utilize super-resolution that takes low-quality video as input and generates an "up-scaled" version. We choose super-resolution because significant advances have been made recently [28, 45, 49]. While we scope our study to desktop-class GPUs and super-resolution, we believe the framework is generic enough to accommodate different types of DNN models and devices.

3 Background and Related Work

Adaptive streaming (e.g., Apples HLS [1], DASH [2]) is designed to handle unpredictable bandwidth variations in the real world. Video is encoded into various bitrates (or resolutions) and divided into fixed length chunks, typically 2 – 10 seconds. An adaptive birate algorithm (ABR) decides the bitrate for each video chunk. Traditional ABR algorithms select bitrates using heuristics based on the estimated network bandwidth [42] and/or the current size of the client-side playback buffer [66]. MPC [77] and Pensieve [52] demonstrate that directly optimizing for the desired QoE objective delivers better outcomes than heuristics-based approaches. In particular, Pensieve uses deep reinforcement learning and learns through "observations" how past decisions and the current state impact the video quality. Oboe [21] dynamically adjusts the ABR parameters depending on the network conditions consulting the offline pre-computation result. Although these algorithms successfully cope with bandwidth variations, they consider neither the effect of client-side quality enhancement nor the dynamics of simultaneously streaming a DNN and video chunks.

Super-resolution recovers a high-resolution image from a single or multiple *lower resolution* image(s). Super-resolution has been used in a variety of computer vision applications, including surveillance [78] and medical imaging [65], where the original high-quality image/video is not available. Recent studies use DNNs [28, 45, 49] to learn low-resolution to high-resolution mapping and demonstrate a significant performance gain over non-DNN approaches [25, 64]. In particular, MDSR [49] is a state-of-the-art DNN that integrates the residual neural network architecture [34] and supports multi-scale inputs. In NAS, we apply super-resolution on top of adaptive streaming to improve user QoE by enhancing low-quality videos at the client side.

Scalable DNN is an emerging type of DNN designed to dynamically adapt to computational resource constraints, enabling *anytime prediction* [36]. A shallow and a deep network are used in resource-constrained and -sufficient environments respectively [24, 36]. ISResNeXt [48] alternatively uses a thin and a wide network that adapts to the width (or channels) of a DNN. Scalable DNN has been applied primarily to image classification/detection tasks. NAS applies anytime prediction to super-resolution and uses it delivering incremental quality enhancement in a streaming context.

DNN-based media compression. Recent studies [18, 61, 71] have shown DNN-based image compression outperforms traditional image codecs, such as JPEG2000 and WebP. The benefit over conventional codecs comes mainly from two aspects: 1) directly optimizing for the target quality metric and 2) adapting the codec configuration based on the image rather than using a fixed configuration [61]. Applying this to video, however, involves significant challenges including the problem of reducing inter-frame redundancy across DNN-encoded images. A recent work [72] performs both I-frame compression and frame interpolation using DNNs. However, the DNN-based video compression is still at its early stage and only offers "comparable performance to MPEG-2" and falls short in delivering real-time decoding [72]. NAS aims to augment existing video delivery using DNN—it applies super-resolution DNNs on top of traditional video codecs by applying quality enhancements frame-by-frame.

Video processing systems. Back-end video processing systems have been of growing importance due to the scale required for video encoding. Studies have reported that latency for fast interactive sharing, system efficiency in encoding, scalability and fault tolerance are major issues [31,37,70]. SVE [37] presents a backend system for video

(a) Original (1080p) (b) Content-aware DNN (c) Content-agnostic DNN (d) 240p

Figure 2: 240p to 1080p super-resolution results
(Content type – 1st row: Game [15], 2nd row: Entertainment [14], 3rd row: News [16])

processing used in Facebook. ExCamera [31] uses massive parallelism to enable interactive and collaborative editing. They focus on solving distributed system problems within a datacenter without changing the clients, whereas we focus on the division of work between the servers and clients.

Studies on video control plane [32, 41, 44, 51] identify spatial and temporal diversity of CDNs in performance and advocate for an Internet-scale control plane which coordinates client behaviors to collectively optimize user QoE. Although they control client behaviors, they do not utilize client computation to directly enhance the video quality.

4 Key Design Choices

Achieving our goal requires redesigning major components of video delivery. This section describes the key design choices we make to overcome practical challenges.

4.1 Content-aware DNN

Key challenge. Developing a universal DNN model that works well across all Internet video is impractical because the number of video episodes is almost infinite. A single DNN of finite capacity, in principle, may not be expressive enough to capture all of them. Note, a fundamental trade-off exists between generalization and specialization for any machine learning approach (i.e., as the model coverage becomes larger, its performance degrades), which is referred to as the 'no free lunch' theorem [75]. Even worse, one can generate 'adversarial' new videos of arbitrarily low quality, given any existing DNN model [38, 58], making the service vulnerable to reduction of quality attacks.

NAS' content-aware model. To tackle the challenge, we consider a content-aware DNN model in which we use a

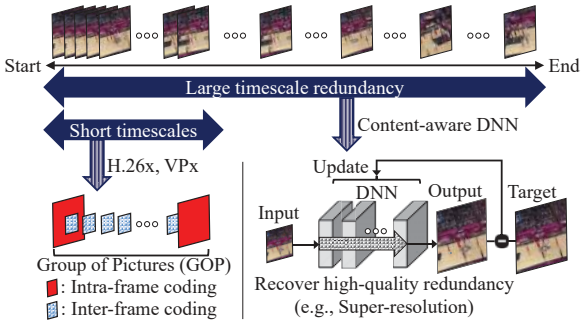


Figure 3: Content-aware DNN based video encoding

different DNN for each video episode. This is attractive because DNNs typically achieve near-zero training error, but the testing error is often much higher (i.e., over-fitting occurs) [67]. Although the deep learning community has made extensive efforts to reduce the gap [40, 67], relying on the DNN's testing accuracy may result in unpredictable performance [38, 58]. NAS exploits DNN's inherent overfitting property to guarantee reliable and superior performance.

Figure 2 shows the super-resolution results of our content-aware DNN and a content-agnostic DNN trained on standard benchmark images (NTIRE 2017 dataset [19]). We use 240p images as input (d) to the super-resolution DNNs to produce output (b) or (c). The images are snapshots of video clips from YouTube. The generic, universal model fails to achieve high quality consistently over a variety of contents—in certain cases, the quality degrades after processing. In §5.1, we show how to design a content-aware DNN for adaptive streaming.

The content-aware approach can be seen as a type of video compression as illustrated in Figure 3. The content-aware DNN captures redundancy that occurs at large time scales (e.g. multiple GOPs) and operates over the entire video. In contrast, the conventional codecs deals with re-

Model name	Compute capacity (Single precision)	Price
GTX 1050 Ti	1.98 TFLOPS	$139
GTX 1060	3.86 TFLOPS	$249
GTX 1070	5.78 TFLOPS	$379
GTX 1070 Ti	7.82 TFLOPS	$449
GTX 1080	8.23 TFLOPS	$559
GTX 1080 Ti	10.61 TFLOPS	$669
Titan Xp	10.79 TFLOPS	$1,200

Table 1: Nvidia's desktop GPU (Geforce 10 series)

dundancy within a frame or between frames within a GOP. In NAS, we demonstrate the new encoding scheme using per-video super-resolution DNNs. However, we believe the content-aware approach can be applied to a series of videos (of similar content) and extended to work with different types of DNNs, such as frame interpolation [56], as discussed in our position paper [76].

4.2 Multiple, Scalable DNNs

Key challenge. The available capacity of computing changes across time and space because of heterogeneity of client devices, changes in workloads, and multiplexing. Table 1 shows even within the desktop-class GPUs the computational power varies up to 5.68 times. Nevertheless, real-time inference is required for online video streaming—the DNN inference has to be at least as fast as the playback rate. However, existing super-resolution DNNs [28, 45, 49] require a fixed amount of computing power and cannot adapt to time-varying capacity. Thus, using a single DNN either under-utilizes client's GPU or does not meet the real-time requirement.

NAS' multiple, scalable DNN design. To tackle the challenge, NAS offers multiple DNNs and let clients dynamically choose one that fits their resource. Similar to multiple bitrates that adaptive streaming offers, we provide a range of DNN options that differ in their inference time (or computational requirements). NAS servers provide multiple DNN specifications as part of the video manifest file. We provide a light-weight mechanism that does not require clients to download the DNNs for choosing the right DNN from available options.

However, using multiple DNNs introduces another challenge. Because the size of DNN grows proportional to its computation requirement, DNNs designed for high-end GPU devices can be very large (a few MBs). It can take a long time to download and utilize the DNN. To address the issue, we design a scalable DNN that enables a client to utilize a partially downloaded model in an incremental fashion. The scalable DNN consists of multiple bypass-able intermediate layers, enabling a partial DNN without the intermediate layers to generate the output as shown in Figure 4. In addition, the design naturally accommodates temporal variation in computational power due to multiplexing. When the computational resource is abundant, clients can use all layers, otherwise they can opportunistically bypass any number of intermediate layers, enabling *anytime prediction* [24, 36, 48]. Finally, the use of multiple scalable DNNs allows each device to benefit from partially downloaded DNNs and provides the same level of temporal adaptation regardless of the device's computational power. §5.2 presents the details of scalable DNNs.

4.3 Integrated ABR

Key challenges. As NAS uses per-video DNN, a client must download a DNN from a server to benefit from DNN-based quality enhancement. However, DNN downloads also compete for the bandwidth with the video stream itself. As a result, aggressively downloading the DNN model may degrade user QoE. At the same time, a client may benefit from an early DNN download because it can receive the quality enhancement early on. Because there exists a conflict, a careful decision making as to when and how to download the DNN is critical.

NAS' bitrate adaptation integrates the decision to download a DNN with bitrate selection for QoE maximization. It considers three additional factors that impact user QoE: 1) To benefit from quality enhancement, a client-side DNN must be downloaded first; 2) a partially downloaded DNN improves quality in proportion to the amount downloaded; and 3) DNN chunk downloads compete for bandwidth with video chunk downloads.

To solve the non-trivial problem, we leverage reinforcement learning [52] and generate an ABR algorithm that integrates the decision to download a DNN model. For this, we divide the DNN model into fixed-size chunks and train an RL network that outputs a decision (i.e., whether to download a video or a DNN chunk) using as input the current state (e.g., throughput measurement, playback buffer occupancy, the number of remained video chunks) and its history. We train the RL network using a large training set consisting of real network traces [9, 60].

The decision to use RL brings a number of benefits: 1) it allows NAS to directly optimize for any target QoE metric, while accounting for the benefit of DNN-based quality enhancement; 2) RL balances multiple interacting factors, such as bandwidth variations, bandwidth sharing between video and DNN chunks, and quality enhancement of partial DNNs, in a way that optimizes the QoE; and 3) it naturally accommodates the use of multiple DNNs by encoding the DNN type in the RL network. §5.3 presents the details of the integrated ABR.

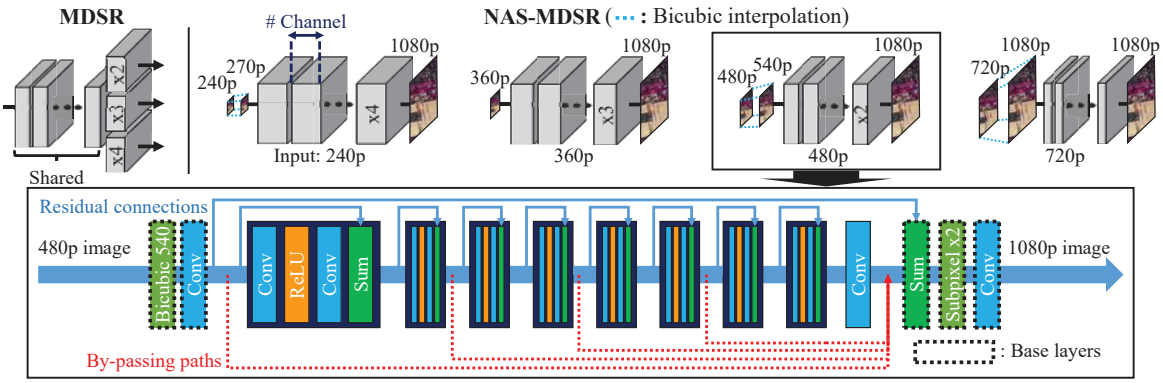


Figure 4: Super-resolution DNN architectures: MDSR vs. NAS-MDSR

5 System Design

NAS is implemented on top of current HTTP adaptive streaming, standardized in DASH [2]. We start by explaining the key differences in how the system operates.

New video admission (server-side processing). As in DASH, when a video clip is uploaded, it is encoded at multiple bitrates and divided into chunks. In addition, the server trains content-aware DNNs for the video for client-side quality enhancement (§5.1). It then associates the DNNs with the video by placing their URLs in the manifest file along with DNN specifications (§5.2).

Client behavior. A client's video player first downloads a manifest file, which contains a list of available DNNs for the video. The client then selects one of them that fits its computing power. The client's DNN processor uses a light-weight mechanism to choose the best available one that fits its resource (§5.2). The player then downloads the selected DNN or video chunks following the decision given by the integrated adaptive bitrate (ABR) algorithm (§5.3). When a DNN chunk is downloaded, the player passes it to the DNN processor, and the processor loads the (partial) DNN on the client's computing device (e.g. GPU). When a video chunk is downloaded, the player keeps it in the playback buffer and the chunk becomes ready for immediate playback. The player then opportunistically passes video chunks in the playback buffer to the DNN processor for quality enhancement along with their associated playback time, which indicates the deadline for neural processing. When the DNN processor finishes processing a chunk, it replaces the original chunk in the playback buffer. Finally, the quality enhanced video chunk is played.

Client-side neural processing. The DNN processor initializes the DNN as DNN chunks arrive. The DNN we use performs quality enhancement on a per-frame basis for super-resolution. Thus, the DNN processor first decodes a video chunk into frames. The DNN processor then applies the super resolution DNN. The resulting frames are then re-encoded to video chunks which replace the original chunks in the playback buffer. The decoding, super-resolution, and encoding phases are pipelined and parallelized to minimize the latency (See §7.5 for details).

5.1 Content-aware DNN for DASH

Applying DNN to adaptive streaming. Standard DNN architectures are not designed for adaptive streaming which introduces specific requirements. First, because adaptive streaming uses multiple resolutions, DNN must be able to take multiple resolutions as input. Second, the DNN inference has to take place in real-time. Finally, DNN should sacrifice its inference quality as little as possible in meeting the first two requirements. The inference time can be reduced at the cost of quality by reducing the number of layers and/or the number of channels (a set of features) in each layer. Such down-scaling also decreases DNN's network footprint. Thus, we must strike a balance between the tradeoff in quality and size, while meeting the real-time requirement.

Using a super-resolution network. Our system extends MDSR [49], a state-of-the-art super-resolution network. As shown in Figure 4, MDSR supports multi-scale super-resolution (x2, x3, x4) in a single network, while sharing the intermediate layers to reduce the footprint. The input resolution drastically affects the inference time of MDSR. For example, even on a flagship desktop GPU (e.g., Nvidia Titan Xp), a 720p input only delivers 3.23 frames per second, whereas a 240p image is processed nearly in real-time at 28.96 frames per second. Thus, to meet the real-time constraint for the highest resolution, one has to downscale the network size.

However, due to the shared layer design, this degrades the quality of all resolutions uniformly, making the lower resolution suffer from significant quality degradation. To avoid the limitation, we use a separate network for each resolution (Figure 4), trading in the total size of the DNN for inference quality. For each DNN, we fix the number

of layers that represents the capacity to adapt to temporal variation in computing power (§5.2). Then, we take the maximum number of channels, independently, for each resolution, that satisfies the real-time constraint. The resulting DNN configuration and size we use for our evaluation are listed in Table 2. The footprint of the 'Ultra-high' DNN is 2,145 KB, which is about the size of a single 1080p (four-second) video chunk (4.8 Mbps) from our evaluation. The size of the 'Low' DNN is only about half the size of a 240p chunk (400 Kbps). While we use NAS-MDSR and specific settings for evaluation, NAS design is not bound to any specific DNNs but accommodates their evolution.

Training content-aware DNNs. The training data is pairs of a low resolution (e.g., 240p, 360p, 480p, 720p) and the original highest resolution (e.g., 1080p) image. We update the DNN parameters to minimize the difference between the DNN's output and the target high resolution image. The training cost is of "one-time" and is amortized across the total watch time of the video episode. Nevertheless, for CDNs that deliver many video episodes, the total computational cost may be high. To reduce the cost of training, we apply the fine-tuning strategy of a transfer learning type. Namely, we first train a generic DNN model using a popular standard benchmark [19]. We then train the content-aware DNN model on each video episode with its weights initialized as those from the generic model. This reduces the computation cost in training by 5-6x, while achieving similar quality enhancement compared to random initialization [33].

5.2 Adaptation to Computational Power

This section describes two mechanisms (multiple DNNs and anytime prediction) that clients use to adapt to their computational capacity to deliver real-time quality enhancement. For each technique, we describe the enabling DNN design and explain the client-side dynamic adaptation logic.

Providing multiple DNNs (server-side). As discussed in §4.2, we provide multiple DNN configurations that vary in quality and computational requirements to support a broad range of GPUs. Similar to GPU's rendering quality options, we provide four quality levels of DNNs: 'Low', 'Medium', 'High', 'Ultra-high'. Thus, we have a DNN per quality per input-resolution, as shown in Table 2. Finally, the server records the available DNN configurations on a manifest file, including the DNN name and level, input resolution, the number of layers, and the number of channels.

Choosing a DNN from multiple options (client-side). Clients test-run the DNN options to choose the one that gives the best quality improvement and delivers real-time performance. A naive way to measure the inference time of DNNs is downloading all DNNs at the client device. However, this consumes large bandwidth (several MBs) and unnecessarily delays video streaming, ultimately degrading user QoE. To streamline the process, NAS provides enough information about the DNN options (i.e., the number of layers and channels) in the manifest file for clients to reconstruct mock DNNs without downloading the DNNs. Using the DNN configuration defined in the manifest file, clients generate DNNs initialized with random weights and run them on their GPUs. Finally, the clients select the largest (highest-quality) DNN that runs in real-time—the client does not need actual weights here because a larger DNN provides better quality. With four DNN options, the client-side test-run takes between 1.64-3.40 seconds depending on a GPU model. Thus, the client can decide which DNN to use early on without downloading any DNN.

Input Resolution	DNN Quality Level			
	Low	Medium	High	Ultra-high
240p	20, 9	20, 21	20, 32	20, 48
	43 KB	203 KB	461 KB	1026 KB
360p	20, 8	20, 18	20, 29	20, 42
	36 KB	157 KB	395 KB	819 KB
480p	20, 4	20, 9	20, 18	20, 26
	12 KB	37 KB	128 KB	259 KB
720p	6, 2	6, 7	6, 16	6, 26
	2 KB	5 KB	17 KB	41 KB

Table 2: DNN configurations for NAS-MDSR (#Layer, #Channel, Size)

Scalable DNN and anytime prediction (server-side). Our scalable DNN architecture enables the client to utilize a partially downloaded DNN and adapt to time-varying computational power. Utilizing partial DNNs provides incremental benefit as the download progresses. This especially benefits the QoE at the beginning of a video because the full DNN of a few MBs cannot be transferred instantly. In addition, the scalable architecture enables anytime prediction allowing us to adapt to client's available computational power that may change unexpectedly.

For this, we modify the DNN architecture and its training method. The DNN's intermediate layers consist of multiple residual blocks [49] each of which consists of two convolutional layers. We allow bypassing consecutive intermediate blocks during inference. To enable bypassing, we add direct connections from intermediate blocks to the final layers, as shown in Figure 4. This creates multiple inference paths as shown in the figure. We then train all interference paths in the following way.

In each training iteration, we randomly bypass intermediate layers to calculate the error between the net-

work output and the target image. Then, we use back-propagation [62] to update parameters. In particular, we go through all layers with probability $1/2$ (for training the original path) and choose one of the remaining by-passing paths uniformly at random otherwise. The resulting DNN can generate an output image using only a part of the DNN and provide incremental quality improvement as more layers are used. Finally, DNNs are divided into chunks. Our server places the chunks' URLs in the video manifest file. The first DNN chunk consists of the base layers for all input resolutions. The subsequent chunks contain the rest of DNNs.

Using the scalable DNN (client-side). A client downloads the DNN in chunks. It reconstructs the first *partial* DNN after downloading the base layers. The size of this minimal DNN is only 35.11% – 36.14% of the full DNN (33 KB to 768 KB), allowing the client to start benefiting from DNN-based quality enhancement early on. As the client downloads more blocks, it updates the DNN, which provides incremental benefit.

Finally, our client opportunistically determines the number of layers to use during video playback. At every time interval, the client's DNN processor first calculates the amount of time remaining until the playback time of the chunk it is processing. The client then calculates the maximum amount of layers it can use that meets the deadline. To aid this, the client records the latest inference time for each layer and updates this table when the inference time changes. We empirically set the time interval to four seconds, which is the length of a single video chunk in our evaluation. This allows NAS clients to dynamically adapt to changes in the available computational power, as we demonstrate in §7.4.

5.3 Integrated Bitrate Adaptation

NAS integrates two decisions into its ABR algorithm for QoE optimization: 1) it decides whether to fetch a video chunk or a DNN chunk; and 2) if the first decision is to fetch a video chunk, it chooses the chunk's bitrate.

The algorithm must balance the two conflicting strategies. The first strategy places emphasis on downloading the DNN model in the hope that this will bring quality enhancement in the future, while sacrificing video's streaming quality at the moment. The second strategy optimizes for video bitrate at the moment and delays the DNN download. In practice, the resulting outcome is unpredictable because it depends on how the network conditions change. The solution space is extremely large considering the number of bitrates, the download order of video and DNN chunks, and the dynamic range of available bandwidth.

To tackle the challenge, we use a reinforcement learning (RL) framework [52,53] that directly optimizes the target metric (without using explicit decision labels) through comprehensive "experience". In particular, we adopt the actor-critic framework of A3C [53]. It learns a strategy (or policy) from observations and produces a mapping from raw observations, such as the fraction of DNN model downloaded, the quality improvement due to DNN, network throughput samples, and playback buffer occupancy, to the decisions described above.

RL design. An RL agent interacts with an environment [68]. For each iteration t, the agent takes an action a_t, after observing a state s_t from the environment. The environment then produces a reward r_t and updates its state to s_{t+1}. A policy is defined as a function that gives the probability of taking action a_t given s_t, $\pi(s_t, a_t) :\rightarrow [0,1]$. The goal then is to learn a policy, π, that maximizes the sum of future discounted reward $\sum_{t=0}^{\infty} \gamma^t r_t$, where $\gamma \in (0,1]$ is a discount-rate for future reward.

In our case, the set of actions $\{a_t\}$ includes whether to download a DNN chunk or to download a video chunk of a specific bitrate. The state s_t includes the number of remaining DNN chunks to download, throughput measurements, and player measurements (e.g., the playback buffer occupancy, past bitrates). Table 3 summarizes the state s_t. The reward r_t is the target QoE metric which is a function of bitrate utility, rebuffering time, and smoothness of selected bitrates [52, 77] defined as:

$$\frac{\sum_{n=1}^{N} q(R_n) - \mu \sum_{n=1}^{N} T_n - \sum_{n=1}^{N-1} \left| q(R_{n+1}) - q(R_n) \right|}{N} \quad (1)$$

where N is the number of video chunks; R_n and T_n respectively represent the video chunk n's bitrate and the rebuffering time resulting from its download; μ is the rebuffering penalty; and $q(R_n)$ is the perceived quality of bitrate R_n (refer to Table 5 in §7.1 for the choices of μ and $q(R_t)$).

To reflect the DNN-based quality enhancement of NAS, we define effective bitrate $R_{\texttt{effective}}$ instead of the nominal bitrate R_n. For each video chunk C_n:

$$R_{\texttt{effective}}(C_n) = \text{SSIM}^{-1}(\text{SSIM}(\text{DNN}_m(C_n)))$$

where $\text{DNN}_m(C_n)$ represents the quality enhanced video chunk C_n after downloading the (partial) DNN chunk m, SSIM is the average structural similarity [73] for measuring the video quality, and its inverse SSIM^{-1} maps a SSIM value back to the video bitrate. To create the mapping, we measure the SSIM of original video chunks at each bitrate (or resolution) and use piece-wise linear interpolation (e.g., (400 Kbps, SSIM_1), ..., (4800 Kbps, SSIM_5)).

Type	State
DNN status	# of remaining DNN chunks
Network status	Throughput for past N chunks
	Download time past N chunks
Player status	Playback buffer occupancy
	Next video chunk sizes
Video status	Bitrate of the latest video chunk
	# of remaining video chunks

Table 3: State used in our RL framework. We use N = 8 which empirically provides good performance.

RL training. Our RL framework has two neural approximators: an actor representing the policy and a critic used to assess the performance of the policy. We use the *policy gradient method* [69] to train the actor and critic networks. The agent first generates trajectories following the current policy $\pi_\theta(s_t, a_t)$, where θ represents parameters (or weights) of the actor's neural network. The critic network observes these trajectories and learns to estimate the *action-value function* $Q^{\pi_\theta}(s_t, a_t)$, which is the total expected reward with respect to taking action a_t starting at state s_t and following the policy π_θ. At each iteration, the actor network uses this estimation to update the model:

$$\theta \leftarrow \theta + \alpha \sum_t \nabla_\theta \log \pi_\theta(s_t, a_t)(Q^{\pi_\theta}(s_t, a_t) - V^{\pi_\theta}(s_t)),$$

where $V^{\pi_\theta}(s_t)$ is the *value function* representing the total expected reward of π_θ starting at state s_t, and α is the learning rate. In our RL framework, because the reward reflects the average QoE enhancement that content-aware DNN delivers, the critic network learns to estimate the updated total reward. This enables the actor network to learn a policy that balances video and DNN downloads to maximize the QoE.

We use a chunk-level simulator similar to that of Pensieve to accelerate the ABR training. It takes network throughput traces and simulates NAS' video streaming dynamics. In addition, we pre-compute the DNN-based quality enhancement by averaging it over all the videos for each DNN quality. We then use the values to produce a generic ABR model.

When a DNN is downloaded, the simulator updates the amount of downloaded DNN chunks (i.e., decrements the state 'number of remaining DNN chunks'). When a video chunk is downloaded, it adds the chunk to the playback buffer. It then computes the QoE that reflects DNN-based quality enhancement, using the (effective) bitrate utility of each chunk and the rebuffing time. Note, the simulator performs neither actual video downloads nor DNN inferences. Thus, it reduces the training time by 97.12% compared to real-time emulation.

Component	Lines of code (LoC)	Changed
DASH video player	19K lines of JavaScript	8.8% (1763)
Content-aware DNN	6.3K lines of Python	- (6.3K)
Integrated ABR algorithm	5.5K lines of Python	- (5.5K)

Table 4: NAS implementation (Lines of Code)

```
<DNN>
    <Representation quality="low">
      <SegmentTemplate
          DNN="$RepresentationQuality$/$Number$"
                startNumber="1" endNumber="5"/>
    </Representation>
      ....
</DNN>
```

Figure 5: NAS manifest file structure

6 Implementation

We implement NAS client by extending a DASH video player. Both the server-side (training) and client-side (inference) DNN processing are implemented using Pytorch [59]. Table 4 shows the lines of code (LoC) for each component.

NAS client (DASH video player). To implement NAS client, we modify dash.js [4] (version 2.4), a reference implementation of MPEG DASH client written in JavaScript. We run the integrated ABR and content-aware DNNs as separate processes. dash.js is configured to fetch the ABR decisions and quality enhanced chunks through inter-process communication. We add DNN metadata on a manifest file as shown in Figure 5. The `quality` attribute indicates the DNN quality level. The `DNN` attribute of `SegmentTemplate` is used to create the chunk URL, and `startNumber` and `endNumber` indicate the chunk index range. In addition, the manifest file includes the number of layers and the number of channels for each DNN.

Training content-aware DNNs. We implement the scalable super-resolution DNN using Pytorch. For training the DNN model, we use input image patches of size 41x41 pixels by randomly cropping the low-resolution images (e.g., 240p, 360p, 480p, 720p) and run the popular ADAM algorithm [46] to optimize DNN parameters. The mini-batch size, weight decaying parameter, and learning rate are set to 64, 10^{-3}, and 10^{-4}, respectively. We initialize the DNN model using parameters of the generic model (§4.1). We then fine-tune it over 100 mini-batch updates per minute of video to generate a content-aware DNN. Finally, we round off its parameters from single-precision (32-bit) to half-precision (16-bit), which halves the DNN size while introducing minimal performance degradation (virtually no difference in SSIM).

Training integrated ABR. We implement our integrated ABR extending Pensieve's implementation [8]. The initial learning rates of actor/critic networks are set to 10^{-4} and

QoE type	Bitrate utility ($q(R)$)	Rebuffer penalty (μ)
$\text{QoE}_{\texttt{lin}}$	R	4.3
$\text{QoE}_{\texttt{log}}$	$log(R/R_{min})$	2.66
$\text{QoE}_{\texttt{hd}}$	0.4→1, 0.8→2, 1.2→3 2.4→12, 4.8→15	8

Table 5: QoE metrics used for evaluation

10^{-3}, respectively. The entropy weight is initialized as 2. We iterate training over 60,000 epochs in which we decay the entropy weight from 2 to 0.055 exponentially over every epoch. Finally, we select the ABR network that delivers the highest QoE for our training traces.

7 Evaluation

We evaluate NAS by answering the following questions.

- How does NAS perform compared to its baseline competitors, and what is the training cost?
- How does each design component of NAS contribute to the overall performance?
- Does NAS effectively adapt to heterogeneous devices and temporal variance in client's computing power?
- What is the end-to-end processing latency and resource usage of NAS?

7.1 Methodology

Videos. We use videos from popular channels on Youtube. For each of the nine Youtube channel categories, we select three popular channels in the order of appearance. We then pick the most popular video from each channel that supports 1080p quality and whose length is longer than 5 minutes—the number of views of 27 video clips ranges from 7M to 737M. Finally, we download 1080p videos and produce multi-bitrate videos following the Youtube and DASH recommendations [3,13]: Each 1080p video is re-encoded using the H.264 codec [11] in which GOP (or chunk size), frame rate, and bitrates are respectively set to 4 seconds, 24 fps and {400, 800, 1200, 2400, 4800}Kbps (which represent for {240, 360, 480, 720, 1080}p resolution videos). Unless otherwise noted, we use the entire video for training and use the first 5 minutes for playback. Training over the entire video ensures that NAS delivers consistent quality enhancement over the entire video.

Network traces. We use a real bandwidth dataset consisting of 508 throughput traces from Norway's 3G network (2010-2011) [60] and 421 traces from U.S. broadband (2016) [9], compiled by the Pensieve author [8]. We filter out the traces that consistently experience low bandwidth ($<$ 400 Kbps) for an extended time ($\geq$100 seconds). The resulting average throughput ranges from 0.38 Mbps to 4.69 Mbp, and the mean and median are 1.31 Mbps and 1.09 Mbps, respectively. Each trace spans 320 seconds, and we loop the trace until a video is completely downloaded. We use randomly selected 80 % of our traces for training and the remaining 20 % for testing.

Baseline. We compare NAS against the following state-of-the-art bitrate adaptation that does not utilize client computation.

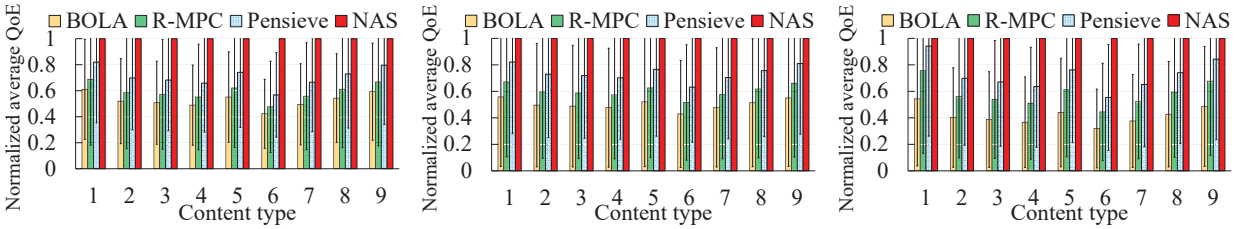
- Pensieve [52] uses deep reinforcement learning to maximize QoE.
- RobustMPC [77] uses playback buffer occupancy and throughput predictions over next five chunks to select the bitrate that maximizes QoE. We use the version reproduced by the authors of Pensieve [8].
- BOLA [66] uses Lyapunov optimization based on playback buffer occupancy. We use the BOLA version implemented in dash.js, which is a Javascript-based reference implementation of a MPEG-DASH player [4].

QoE metrics. We use three types QoE metrics, compiled by MPC and Pensieve, whose the bitrate utility function, $q(R_n)$, and rebuffering penalty constant, μ of Equation 1, differ as summarized in Table 5.

- $\text{QoE}_{\texttt{lin}}$ uses a linear bitrate utility.
- $\text{QoE}_{\texttt{log}}$ uses a logarithmic bitrate utility function that represents its decreasing marginal utility.
- $\text{QoE}_{\texttt{hd}}$ heavily favors high-definition (HD) video (720p and 1080p) over non-HD.

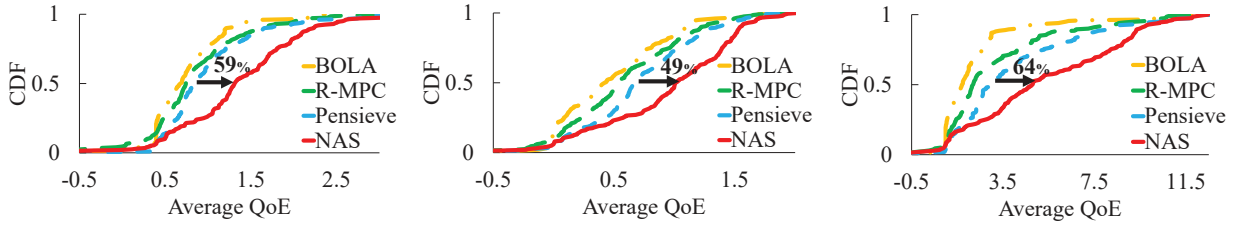
Experimental settings. We run our dash.js implementation on a Chromium Browser (version 65) to stream MPEG-DASH videos. We use six GPU models from Nvidia's desktop GPU product line listed in Table 1. Unless otherwise noted, the default client-side GPU is Nvidia Titan Xp. In our setting, the content-aware DNN and the ABR network run at the client as separate processes. To emulate the network conditions from the network traces, we use Mahimahi [55].

We use two experiment settings. To evaluate NAS client on all six GPUs, we have a local testbed. To scale training and testing, we use Google Cloud Platform. Training is done using GPU instances equipped with Nvidia's server-class Tesla P100 GPU. However, Google Cloud Platform does not have desktop class GPUs, while we need to scale client-side streaming experiments to 18 hours of network traces x 27 video clips x 4 types of ABR x 3 types of QoE, totaling 5,832 hours of streaming time. Thus, we take quality and latency measurements of content-aware DNNs using the local testbed on each GPU device for each video. We then emulate the network condition between NAS server and client once for each network trace and apply the effect of the quality enhancement and latency of content-aware DNNs. We confirm the network-emulated, DNN-simulated clients produce



(a) QoE_{lin} (b) QoE_{log} (c) QoE_{hd}

Figure 6: Normalized QoE comparison of video clips from the nine content categories of YouTube. (1: Beauty, 2: Comedy, 3: Cook, 4: Entertainment, 5: Game, 6: Music, 7: News, 8: Sports, 9: Technology)



(a) QoE_{lin} (b) QoE_{log} (c) QoE_{hd}

Figure 7: Cumulative distribution of QoE for 'Sports' content category

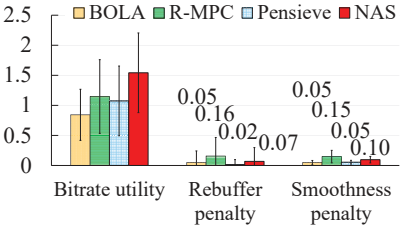


Figure 8: QoE_{lin} breakdown

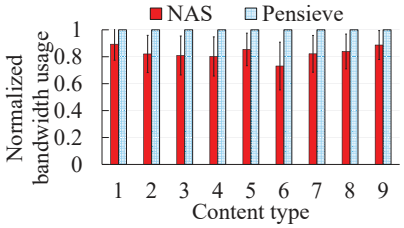


Figure 9: Normalized bandwidth usage at QoE_{lin}=0.98

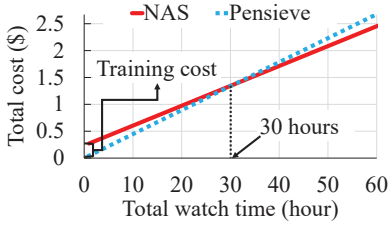


Figure 10: Cumulative server cost

the same QoE as real clients of our local testbed using a fraction of test data.

7.2 NAS vs. Existing Video Delivery

QoE improvement. Figure 6 shows the average QoE of video clips across the nine content categories. The error bars indicate one standard deviation from the average. NAS delivers the highest QoE across all content categories over all three QoE metrics. The result shows significant improvement over prior work. NAS consistently outperforms Pensieve by a large margin across all QoE metrics: QoE_{lin} (43.08% better), QoE_{log} (36.26% better), and QoE_{hd} (42.57% better). With QoE_{lin}, NAS outperforms Pensieve 43.08% on average, whereas Pensieve achieves a 19.31% improvement over RobustMPC (R-MPC). Compared to BOLA, NAS achieves 92.28% improvement in QoE_{lin}. The QoE improvement varies across content types from 21.89% (1: 'Beauty') to 76.04% (6: 'Music') over Pensieve because many factors, such as the scene complexity, compression artifacts, and temporal redundancy, affect the DNN performance.

Figure 7 shows the cumulative distribution of QoE over our test traces. It shows the 'Sports' content category which shows medium gain among all categories. NAS delivers benefit across all network conditions. NAS improves the median QoE_{lin} by 58.55% over Pensieve. Note, Pensieve mainly delivers its QoE gain over RobustMPC by reducing rebuffering at the cost of bitrate utility. In contrast, NAS does not exhibit such tradeoff because it uses client computation. Other content (not shown) displays a similar trend. Finally, Figure 8 shows a breakdown of QoE into bitrate utility, rebuffering penalty, and the smoothness penalty. NAS benefits the most from the bitrate utility due to the DNN-based quality enhancement.

Bandwidth savings. Despite the DNN transfer overhead, NAS requires less bandwidth in delivering the same QoE level. To demonstrate this, we create a hypothetical setting using the chunk-level simulator (§5.3) where NAS clients receive a fraction of bandwidth that Pensieve clients receive including the DNN transfer overhead. We adjust the fraction and empirically determine the fraction that deliv-

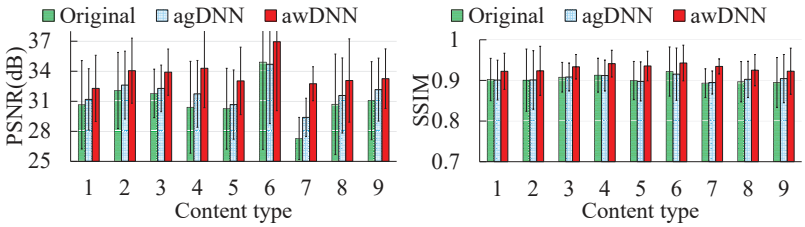


Figure 11: Video quality in PSNR and SSIM (240p input)

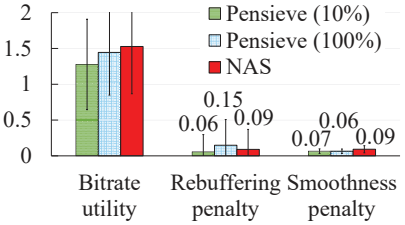
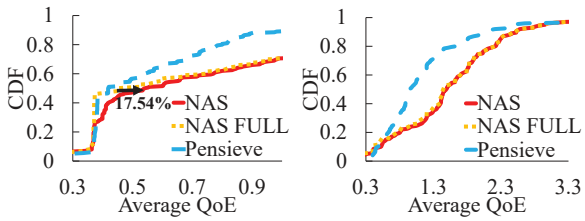


Figure 12: Integrated ABR vs. Baseline algorithm



(a) $t \leq 47.82$ seconds (b) $t > 47.82$ seconds

Figure 13: Scalable DNN vs. Full DNN

ers the same QoE. We assume NAS clients download the largest DNN ('Ultra-high') model for every five-minute video. Figure 9 shows the average bandwidth usage of Pensieve and NAS. On average across all videos, NAS requires 17.13% less bandwidth than Pensieve to deliver the same quality. The savings vary across content between 10.69% and 26.90%. This demonstrates the benefit of using a DNN outweighs the overhead of transferring the DNN.

Cost-benefit analysis (server-side). We now quantify the overall server-side cost in using a NAS content delivery network. While NAS servers use less bandwidth to deliver the same quality, they must train content-aware DNNs and the integrated ABR network. We quantify the computation and bandwidth cost of the CDN servers. The training time for the integrated ABR is only 10.92 hours on a CPU. Because it is a one-time cost amortized across all video streams, the additional cost is negligible. In contrast, the content-aware DNNs must be trained for each video. The total training time (across multiple DNNs) per minute of video is 10 minutes.

For a Google cloud instance with 8 vCPUs, 32 GB RAM, and a Nvidia P100 GPU, this translates to $0.23 per minute of video. For bandwidth, Amazon CDN instance charges at most 0.085 $/GB. The price per bandwidth becomes cheaper as one uses more bandwidth. Using these as reference, we compute the total video delivery cost a function of cumulative viewing time per minute of video. Figure 10 shows the cost comparison for NAS and Pensieve. As before, we assume each user watches a video clip for five minutes (i.e., DNNs are transferred every five minutes of viewing). This is a conservative estimate given the popularity of binge-watching [5]. NAS pays the upfront cost of computation, but as the cumulative viewing time increases, it is amortized. Note, NAS uses 17.13% less bandwidth to deliver the same user QoE. Thus, when the cumulative viewing reaches 30 hours (per minute of video in the system), NAS CDN recoups the initial investment.

7.3 Component-wise Analysis

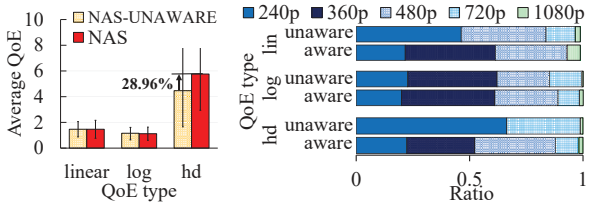
We evaluate how each design component contributes to the quality improvement.

Content-awareness. Figure 11 compares video quality of content-aware DNN (awDNN), a content-agnostic DNN (agDNN) trained on standard benchmark images (NTIRE 2017 dataset [19]), and the original 240p video we use as input upscaled by the bicubic interpolation. We measure the video quality both in PSNR [35] and SSIM [73] in which PSNR represents the average mean square error between two images in logarithmic decibel scale. Content-aware DNN delivers consistent improvement whereas content-agnostic DNNs even degrades the quality in some cases with respect to the PNSR measure (content type: 6) and the SSIM measure (type: 1,2,4,5,6). This confirms our rationale for using DNN's training accuracy.

Scalable DNN. Figure 13 demonstrates the benefit of utilizing a partial DNN. We compare Pensieve, NAS, and a version of NAS (`NAS-FULL`) that does not utilize partial DNN downloads. Specifically, Figure 13(a) shows the cumulative distribution of QoE_{lin} before the average full DNN download time (47.82 seconds). As soon as a partial DNN is downloaded (22.16 seconds on average), NAS enhances the quality. The result shows that this delivers 17.54 % and 3.35 % QoE improvement in the median and mean, respectively. Note, the QoE of NAS and `NAS-FULL` becomes identical after downloading the full DNN ($t > 47.82$ seconds) as shown in Figure 13(b).

Integrated ABR. The integrated ABR delivers benefit during and after the DNN download. To demonstrate this, we create two hypothetical settings using the chunk-level simulator (§5.3).

First, we compare NAS with a version that uses a naive



(a) Average QoE (b) Video quality distribution

Figure 14: Integrated ABR vs. Quality-unaware ABR

Model Name	Frames per second (FPS)			
	Low	Medium	High	Ultra-high
GTX 1050 Ti	**34.36**	17.62	14.91	7.37
GTX 1060	45.27	**30.05**	25.96	13.17
GTX 1070 Ti	41.76	45.24	**41.53**	21.47
GTX 1080	53.82	52.86	**38.95**	21.46
GTX 1080 Ti	58.94	56.29	57.12	**31.34**
Titan Xp	52.99	51.72	52.22	**33.58**

Table 6: DNN processing speed on desktop GPUs (Bold font indicates the selected quality level.)

DNN download strategy that downloads a DNN at a fraction of the video bitrate chosen by Pensieve. Note, it does not integrate the bitrate selection and DNN download decision. We use two variants: one that aggressively downloads DNN at 100% of the video bitrate and the other that uses only 10%. Both are configured to start downloading the DNN when the playback buffer becomes larger than 15.42 seconds, which is the average time that NAS starts to stream a DNN in our test traffic traces. Our result shows NAS respectively outperforms the non-aggressive and aggressive strawman by 16.36% and 9.13% with respect to QoE_{lin}. Figure 12 shows the comparison of QoE components. The non-aggressive version experiences lower bitrate utility compared to NAS because the former downloads the DNN more slowly. In contrast, the aggressive version increases the rebuffering penalty by x2.5 which negatively affects the QoE.

Next, to evaluate the benefit of quality-enhancement aware bitrate selection after the DNN is fully downloaded, we compare NAS with a quality-enhancement unaware ABR after the full DNN download. Figure 14(a) shows the average QoE in this setting. We see that the quality-enhancement aware ABR delivers a large gain for QoE_{hd} (28.96 %), whereas it offers minimal benefit to QoE_{lin} (0.01 %) and slightly degrades on QoE_{log} (-3.14 %). The reason it delivers a large gain for QoE_{hd} is because the DNN-enhanced quality of 1.2 Mbps (480p) videos get close to that of the original 2.4 Mbps (720p) video and the marginal utility with respect to the increased quality is far greater for QoE_{hd} than any other QoE type, especially between 1.2 Mbps to 2.4 Mbps (Table 5). The integrated

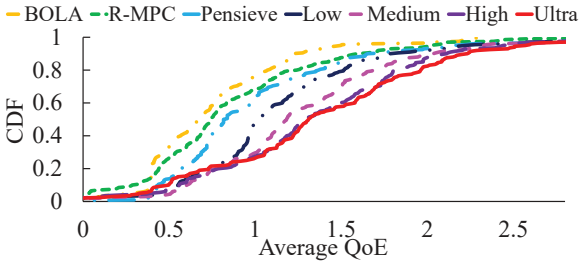


Figure 15: CDF of QoE over different quality DNNs

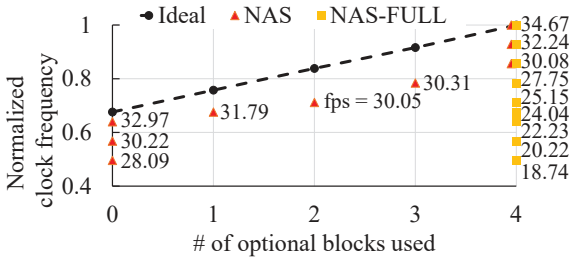


Figure 16: Dynamic adaptation to computing power

ABR reflects this enhancement in the bitrate decisions to download 480p much more often when using QoE_{hd} as shown in Figure 14(b).

7.4 Dynamic Adaptation to Computation

We demonstrate NAS's ability to adapt to heterogeneous clients and temporal variation in computing power.

Heterogeneous clients. We demonstrate NAS is able to meet real-time constraints on six desktop GPUs shown in Table 1. We run our DASH client on six clients each with a different GPU model and measure their performance. First, we measure the throughput of the DNN processing engine, which includes decoding, DNN inference, and re-encoding. Table 6 reports the minimum processing speed across all input resolutions for each device. The video playback rate is 30 frames per second. Clients perform a test-run when it receives a video manifest file. The selected quality level (e.g., 'Low', 'Medium', 'High', or 'Ultra-high') is indicated in boldface. We see that each device selects one that meets the real-time constraint. Note the processing time does not depend on video content.

Next, we measure the QoE of clients using four different quality levels in our cloud setting. Figure 15 shows the cumulative distribution of QoE_{lin} for each quality level. All quality levels outperform Pensieve. The higher the quality level DNN, the better the quality it delivers. Note, even though DNNs of higher quality are larger in size, they deliver incremental benefit over lower quality DNNs.

In sum, the results indicate that NAS adapts to heterogeneous devices, and a device with higher computational power receives greater benefit.

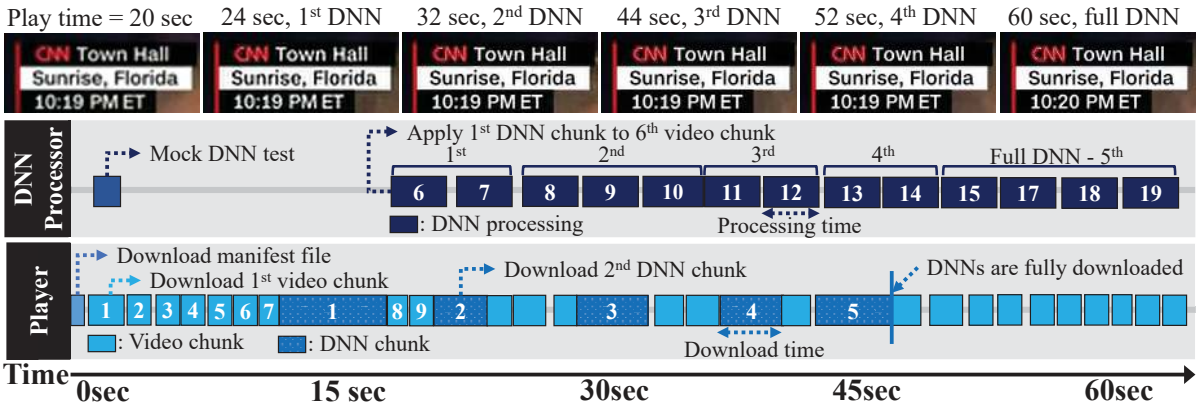


Figure 17: Case study: A time-line of NAS client in operation (Video Source: [16])

Phase	Processing time (sec)
Decode	0.28
Super resolution	3.69
Encode	0.91
Total	4.88
NAS' parallel pipeline	3.76 (23.0% reduction)

Table 7: Video processing time per phase

Temporal variation. We evaluate client's adaptation to temporal variation in computing power in isolation. We emulate the changes in available computing power by varying the clock frequency of GPU (Titan Xp). We change the GPU clock frequency at 10% granularity and report the inference path used and its throughput. Figure 16 shows the result compared to a naive version that only uses the full DNN (`NAS-FULL`) and the ideal line that plots the normalized throughput (y-axis) of each inference path at full clock cycle. x-axis shows the number of optional blocks used for inference. The y-intercept represents the computing requirement for the required layers of DNN. We report the raw throughput for all results. It shows NAS adapts to the changing resource with anytime prediction to the extent that the underlying DNN supports and thus delivers real-time performance unlike `NAS-FULL` that does not.

7.5 End-to-end Operation

NAS client in operation. We present an end-to-end operation of our NAS client using a network trace from our testset. We use a client with a Titan Xp GPU, running on our local testbed. Figure 17 shows the time-line starting from a video request made from our DASH client. At $t = 0.36$ (sec), it downloads the video manifest and test-runs the mock DNNs to select the DNN quality level. The test-run finishes at $t = 2$. The video starts playing at $t = 2.03$. At $t = 17.64$, the first DNN chunk is downloaded, and the minimal DNN initialized at $t = 17.71$. At this time, the DNN processing begins, and video chunks (6-7) in the playback buffer receive quality enhancement. Subsequent video chunks are processed by the DNN as they arrive. As new DNN chunks arrive, the DNNs are incrementally updated. At $t = 46.41$, DNNs are fully downloaded.

DNN processing time. We evaluate the DNN processing latency of NAS client. For DNN processing, NAS pipelines three processes, each of which respectively handles decoding, super-resolution, and re-encoding. Table 7 shows the processing time for each phase for a four-second video chunk. We use GTX 1080 Ti for processing 'Ultra-high' quality DNN using a 30 fps, 240p video as input. We re-encode the DNN's output in H.264 using the fastest option in ffmpeg [6]. This is because the compression factor is not important. The total processing time when each phase is serialized is 4.88 seconds, whereas our pipelined processing takes 3.76 seconds. Considering super-resolution takes 3.69 seconds, the latency overhead of the rest is minimal.

Finally, we measure the client's GPU memory usage for DNN processing. 'Ultra-high', 'High', 'Medium', 'Low' quality DNNs respectively use 3.57 GB, 3.12 GB, 3.05 GB, and 2.99 GB of GPU memory.

8 Conclusion

We present NAS, a video delivery system that utilizes client computation to enhance the video quality. Unlike existing video delivery that solely relies on the bandwidth resource, NAS uses client-side computation powered by deep neural networks (DNNs). NAS introduces new system designs to address practical problems in realizing the vision on top of DASH. Our evaluation over real videos on real network traces shows NAS delivers improvement between 21.89–76.04% in user quality of experience (QoE) over the current state of the art. Finally, the cost-benefit analysis shows content distribution networks can actually reduce the cost of video delivery while providing the same or better QoE compared to the current state of the art.

9 Acknowledgments

We thank our shepherd Wyatt Lloyd and anonymous reviewers for their constructive feedback. This work was supported by Institute for Information & communications Technology Promotion (IITP) grant funded by the Korea government (Ministry of Science and ICT) [No.2018-0-00693] and [No.R-20160222-002755]. Dongsu Han is the corresponding author.

References

[1] Apple HTTP live streaming official homepage. https://developer.apple.com/streaming/.

[2] Dash industry forum. https://dashif.org/.

[3] Dash recommend segment length. https://bitmovin.com/mpeg-dash-hls-segment-length/.

[4] dash.js Github repository. https://github.com/Dash-Industry-Forum/dash.js.

[5] Deloitte: 73 Percent of Americans Binge Watch TV; Millennial Binge Watchers Average Six Episodes and Five Hours per Viewing. https://www.prnewswire.com/news-releases/deloitte-73-percent-of-americans-binge-watch-tv-millennial-binge-watchers-average-six-episodes-and-five-hours-per-viewing-300427152.html.

[6] FFmpeg - H.264 Video Encoding Guide. https://trac.ffmpeg.org/wiki/Encode/H.264.

[7] The iPhone X's new neural engine exemplifies Apple's approach to AI. https://www.theverge.com/2017/9/13/16300464/apple-iphone-x-ai-neural-engine.

[8] Pensieve official Github repository. https://github.com/hongzimao/pensieve.

[9] Raw Data - Measuring Broadband America 2016. https://www.fcc.gov/reports-research/reports/measuring-broadband-america/raw-data-measuring-broadband-america-2016.

[10] The state of online video 2017. https://www.limelight.com/resources/white-paper/state-of-online-video-2017/#weeklyconsumption.

[11] Video codec - H.264 standardization. https://www.itu.int/rec/T-REC-H.264/.

[12] Video codec - H.265 standardization. http://x265.org/.

[13] Youtube recommended upload encoding settings. https://support.google.com/youtube/answer/1722171?hl=en.

[14] Everything Wrong With Frozen In 10 Minutes Or Less. https://www.youtube.com/watch?v=HvwMtWkfkJ8, June 2014.

[15] Minecraft Xbox - School Day [244]. https://www.youtube.com/watch?v=5N6E2cF-CGw, Nov. 2014.

[16] Shooting survivor confronts NRA spokesperson Dana Loesch. https://www.youtube.com/watch?v=4AtOU0dDXv8, Feb. 2018.

[17] ADHIKARI, V. K., GUO, Y., HAO, F., HILT, V., ZHANG, Z. L., VARVELLO, M., AND STEINER, M. Measurement study of netflix, hulu, and a tale of three cdns. *IEEE/ACM Transactions on Networking (ToN) 23*, 6 (Dec 2015), 1984–1997.

[18] AGUSTSSON, E., MENTZER, F., TSCHANNEN, M., CAVIGELLI, L., TIMOFTE, R., BENINI, L., AND GOOL, L. V. Soft-to-hard vector quantization for end-to-end learning compressible representations. In *Advances in Neural Information Processing Systems* (2017), pp. 1141–1151.

[19] AGUSTSSON, E., AND TIMOFTE, R. Ntire 2017 challenge on single image super-resolution: Dataset and study. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops* (July 2017).

[20] AKHSHABI, S., BEGEN, A. C., AND DOVROLIS, C. An experimental evaluation of rate-adaptation algorithms in adaptive streaming over http. In *Proceedings of the ACM Multimedia Systems Conference (MMSys)* (2011), ACM, pp. 157–168.

[21] AKHTAR, Z., NAM, Y. S., RAO, S., AND RIBEIRO, B. Oboe : Auto-tuning video abr algorithms to network conditions. In *Proceedings of the ACM Special Interest Group on Data Communication (SIGCOMM)* (2018).

[22] BALACHANDRAN, A., SEKAR, V., AKELLA, A., SESHAN, S., STOICA, I., AND ZHANG, H. Developing a predictive model of quality of experience for internet video. In *Proceedings of the ACM Special Interest Group on Data Communication (SIGCOMM)* (2013).

[23] BENGIO, Y. Learning deep architectures for AI. *Foundations and Trends in Machine Learning 2*, 1 (2009), 1–127.

[24] BOLUKBASI, T., WANG, J., DEKEL, O., AND SALIGRAMA, V. Adaptive neural networks for efficient inference. In *Proceedings of the International Conference on Machine Learning (ICML)* (2017), pp. 527–536.

[25] CHANG, H., YEUNG, D.-Y., AND XIONG, Y. Super-resolution through neighbor embedding. In *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on* (2004), vol. 1, IEEE, pp. I–I.

[26] CISCO. Cisco visual networking index: Forecast and methodology, 20162021. http://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/complete-white-paper-c11-481360.pdf, July 2017.

[27] DOBRIAN, F., SEKAR, V., AWAN, A., STOICA, I., JOSEPH, D., GANJAM, A., ZHAN, J., AND ZHANG, H. Understanding the impact of video quality on user engagement. In *Proceedings of the ACM Special Interest Group on Data Communication (SIGCOMM)* (2011).

[28] DONG, C., LOY, C. C., HE, K., AND TANG, X. Image super-resolution using deep convolutional networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence 38*, 2 (2016), 295–307.

[29] FATAHALIAN, K. The rise of mobile visual computing systems. *IEEE Pervasive Computing 15*, 2 (Apr 2016), 8–13.

[30] FITZEK, F. H. P., AND REISSLEIN, M. Mpeg-4 and h.263 video traces for network performance evaluation. *IEEE Network 15*, 6 (Nov 2001), 40–54.

[31] FOULADI, S., WAHBY, R. S., SHACKLETT, B., BALASUBRAMANIAM, K. V., ZENG, W., BHALERAO, R., SIVARAMAN, A., PORTER, G., AND WINSTEIN, K. Encoding, fast and slow: Low-latency video processing using thousands of tiny threads. In *Proceedings of the USENIX Conference on Networked Systems Design and Implementation (NSDI)* (Berkeley, CA, USA, 2017), USENIX Association, pp. 363–376.

[32] GANJAM, A., SIDDIQUI, F., ZHAN, J., LIU, X., STOICA, I., JIANG, J., SEKAR, V., AND ZHANG, H. C3: Internet-scale control plane for video quality optimization. In *Proceedings of the USENIX Conference on Networked Systems Design and Implementation (NSDI)* (2015), vol. 15, pp. 131–144.

[33] HE, K., ZHANG, X., REN, S., AND SUN, J. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)* (2015), pp. 1026–1034.

[34] HE, K., ZHANG, X., REN, S., AND SUN, J. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (2016), pp. 770–778.

[35] HORE, A., AND ZIOU, D. Image quality metrics: Psnr vs. ssim. In *Pattern recognition (icpr), 2010 20th international conference on* (2010), IEEE, pp. 2366–2369.

[36] HU, H., DEY, D., HEBERT, M., AND BAGNELL, J. A. Anytime neural network: a versatile trade-off between computation and accuracy. *arXiv preprint arXiv:1708.06832* (2018).

[37] HUANG, Q., ANG, P., KNOWLES, P., NYKIEL, T., TVERDOKHLIB, I., YAJURVEDI, A., DAPOLITO, IV, P., YAN, X., BYKOV, M., LIANG, C., TALWAR, M., MATHUR, A., KULKARNI, S., BURKE, M., AND LLOYD, W. Sve: Distributed video processing at facebook scale. In *Proceedings of the ACM Symposium on Operating Systems Principles (SOSP)* (New York, NY, USA, 2017), ACM, pp. 87–103.

[38] HUANG, S., PAPERNOT, N., GOODFELLOW, I., DUAN, Y., AND ABBEEL, P. Adversarial attacks on neural network policies. *arXiv preprint arXiv:1702.02284* (2017).

[39] HUANG, T.-Y., HANDIGOL, N., HELLER, B., MCKEOWN, N., AND JOHARI, R. Confused, timid, and unstable: Picking a video streaming rate is hard. In *Proceedings of the Internet Measurement Conference (IMC)* (2012), pp. 225–238.

[40] IOFFE, S., AND SZEGEDY, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167* (2015).

[41] JIANG, J., SEKAR, V., MILNER, H., SHEPHERD, D., STOICA, I., AND ZHANG, H. Cfa: A practical prediction system for video qoe optimization. In *Proceedings of the USENIX Conference on Networked Systems Design and Implementation (NSDI)* (2016), pp. 137–150.

[42] JIANG, J., SEKAR, V., AND ZHANG, H. Improving fairness, efficiency, and stability in http-based adaptive video streaming with festive. In *Proceedings of the International Conference on emerging Networking EXperiments and Technologies (CoNEXT)* (2012).

[43] JIANG, J., SEKAR, V., AND ZHANG, H. Improving fairness, efficiency, and stability in http-based adaptive video streaming with festive. *IEEE/ACM Transactions on Networking (ToN) 22*, 1 (2014), 326–340.

[44] JIANG, J., SUN, S., SEKAR, V., AND ZHANG, H. Pytheas: Enabling data-driven quality of experience optimization using group-based exploration-exploitation. In *Proceedings of the USENIX Conference on Networked Systems Design and Implementation (NSDI)* (2017), vol. 1, p. 3.

[45] KIM, J., KWON LEE, J., AND MU LEE, K. Accurate image super-resolution using very deep convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2016), pp. 1646–1654.

[46] KINGMA, D. P., AND BA, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[47] KRISHNAN, S. S., AND SITARAMAN, R. K. Video stream quality impacts viewer behavior: inferring causality using quasi-experimental designs. *IEEE/ACM Transactions on Networking (ToN) 21*, 6 (2013), 2001–2014.

[48] LEE, H., AND SHIN, J. Anytime neural prediction via slicing networks vertically, 2018.

[49] LIM, B., SON, S., KIM, H., NAH, S., AND LEE, K. M. Enhanced deep residual networks for single image super-resolution. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops* (2017).

[50] LIU, H. H., WANG, Y., YANG, Y. R., WANG, H., AND TIAN, C. Optimizing cost and performance for content multihoming. In *Proceedings of the ACM Special Interest Group on Data Communication (SIGCOMM)* (2012).

[51] LIU, X., DOBRIAN, F., MILNER, H., JIANG, J., SEKAR, V., STOICA, I., AND ZHANG, H. A case for a coordinated internet video control plane. In *Proceedings of the ACM Special Interest Group on Data Communication (SIGCOMM)* (2012), pp. 359–370.

[52] MAO, H., NETRAVALI, R., AND ALIZADEH, M. Neural adaptive video streaming with pensieve. In *Proceedings of the ACM Special Interest Group on Data Communication (SIGCOMM)* (2017), pp. 197–210.

[53] MNIH, V., BADIA, A. P., MIRZA, M., GRAVES, A., LILLICRAP, T., HARLEY, T., SILVER, D., AND KAVUKCUOGLU, K. Asynchronous methods for deep reinforcement learning. In *Proceedings of the International Conference on Machine Learning (ICML)* (2016), pp. 1928–1937.

[54] MUKERJEE, M. K., NAYLOR, D., JIANG, J., HAN, D., SESHAN, S., AND ZHANG, H. Practical, real-time centralized control for cdn-based live video delivery. In *Proceedings of the ACM Special Interest Group on Data Communication (SIGCOMM)* (2015), pp. 311–324.

[55] NETRAVALI, R., SIVARAMAN, A., DAS, S., GOYAL, A., WINSTEIN, K., MICKENS, J., AND BALAKRISHNAN, H. Mahimahi: Accurate record-and-replay for http. In *Proceedings of the USENIX Annual Technical Conference (ATC)* (2015), pp. 417–429.

[56] NIKLAUS, S., MAI, L., AND LIU, F. Video frame interpolation via adaptive convolution. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2017).

[57] OOYALA. Ooyala global video index q1 2017. `http://go.ooyala.com/rs/447-EQK-225/images/Ooyala-Global-Video-Index-Q1-2017.pdf`. Last accessed: July 2017.

[58] PAPERNOT, N., MCDANIEL, P., JHA, S., FREDRIKSON, M., CELIK, Z. B., AND SWAMI, A. The limitations of deep learning in adversarial settings. In *Proceedings of the IEEE European Symposium on Security and Privacy (EuroS&P)* (2016), IEEE.

[59] PASZKE, A., GROSS, S., CHINTALA, S., CHANAN, G., YANG, E., DEVITO, Z., LIN, Z., DESMAISON, A., ANTIGA, L., AND LERER, A. Automatic differentiation in pytorch. In *Proceedings of the Advances in Neural Information Processing Systems (NIPS) Workshop* (2017).

[60] RIISER, H., VIGMOSTAD, P., GRIWODZ, C., AND HALVORSEN, P. Commute path bandwidth traces from 3g networks: analysis and applications. In *Proceedings of the ACM Multimedia Systems Conference (MMSys)* (2013), pp. 114–118.

[61] RIPPEL, O., AND BOURDEV, L. Real-time adaptive image compression. In *International Conference on Machine Learning* (2017).

[62] RUMELHART, D. E., HINTON, G. E., AND WILLIAMS, R. J. Learning representations by back-propagating errors. *nature 323*, 6088 (1986), 533.

[63] SANDVINE. 2016 global internet phenomena report: North america and latin america. https://www.sandvine.com/downloads/general/global-internet-phenomena/2016/global-internet-phenomena-report-latin-america-and-north-america.pdf. Last accessed: July 2017.

[64] SCHULTER, S., LEISTNER, C., AND BISCHOF, H. Fast and accurate image upscaling with super-resolution forests. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2015), pp. 3791–3799.

[65] SHI, W., CABALLERO, J., LEDIG, C., ZHUANG, X., BAI, W., BHATIA, K., DE MARVAO, A. M. S. M., DAWES, T., OREGAN, D., AND RUECKERT, D. Cardiac image super-resolution with global correspondence using multi-atlas patchmatch. In *International Conference on Medical Image Computing and Computer-Assisted Intervention* (2013), Springer, pp. 9–16.

[66] SPITERI, K., URGAONKAR, R., AND SITARAMAN, R. K. Bola: Near-optimal bitrate adaptation for online videos. In *Proceedings of the IEEE International Conference on Computer Communications (INFOCOM)* (2016), IEEE, pp. 1–9.

[67] SRIVASTAVA, N., HINTON, G., KRIZHEVSKY, A., SUTSKEVER, I., AND SALAKHUTDINOV, R. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research 15*, 1 (2014), 1929–1958.

[68] SUTTON, R. S., AND BARTO, A. G. *Reinforcement learning: An introduction*. MIT press Cambridge, 1998.

[69] SUTTON, R. S., MCALLESTER, D. A., SINGH, S. P., AND MANSOUR, Y. Policy gradient methods for reinforcement learning with function approximation. In *Proceedings of the Advances in Neural Information Processing Systems (NIPS)* (2000), pp. 1057–1063.

[70] TANG, L., HUANG, Q., PUNTAMBEKAR, A., VIGFUSSON, Y., LLOYD, W., AND LI, K. Popularity prediction of facebook videos for higher quality streaming. In *Proceedings of the USENIX Annual Technical Conference (ATC)* (2017).

[71] TODERICI, G., VINCENT, D., JOHNSTON, N., HWANG, S. J., MINNEN, D., SHOR, J., AND COVELL, M. Full resolution image compression with recurrent neural networks. In *International Conference on Machine Learning* (2017).

[72] TSCHANNEN, M., AGUSTSSON, E., AND LUCIC, M. Deep generative models for distribution-preserving lossy compression. *arXiv preprint arXiv:1805.11057* (2018).

[73] WANG, Z., BOVIK, A. C., SHEIKH, H. R., AND SIMONCELLI, E. P. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing 13*, 4 (2004), 600–612.

[74] WENDELL, P., JIANG, J. W., FREEDMAN, M. J., AND REXFORD, J. Donar: Decentralized server selection for cloud services. In *Proceedings of the ACM Special Interest Group on Data Communication (SIGCOMM)* (2010).

[75] WOLPERT, D. H. The lack of a priori distinctions between learning algorithms. *Neural Computation 8*, 7 (1996), 1341–1390.

[76] YEO, H., DO, S., AND HAN, D. How will deep learning change internet video delivery? In *Proceedings of the 16th ACM Workshop on Hot Topics in Networks* (2017), ACM, pp. 57–64.

[77] YIN, X., JINDAL, A., SEKAR, V., AND SINOPOLI, B. A control-theoretic approach for dynamic adaptive video streaming over http. In *Proceedings of the ACM Special Interest Group on Data Communication (SIGCOMM)* (2015).

[78] ZOU, W. W., AND YUEN, P. C. Very low resolution face recognition problem. *IEEE Transactions on Image Processing 21*, 1 (2012), 327–340.