Minimizing Packet Retransmission for Real-Time Video Analytics

Haodong Wang, Kuntai Du, Junchen Jiang
University of Chicago

ABSTRACT

In smart-city and video-analytics (VA) applications, high-quality data streams (video frames) must be accurately analyzed with a low delay. Since maintaining high accuracy requires compute-intensive deep neural nets (DNNs), these applications often stream massive video data to remote, more powerful cloud servers, giving rise to a strong need for *low streaming delay between video sensors and cloud servers while still delivering enough data for accurate DNN inference.* In response, many recent efforts have proposed distributed VA systems that aggressively compress/prune video frames deemed less important to DNN inference, with the underlying assumptions being that (1) without increasing available bandwidth, reducing delays means sending fewer bits, and (2) the most important frames can be precisely determined before streaming. This short paper challenges both views. First, in high-bandwidth networks, the delay of real-time videos is primarily bounded by packet losses and delay jitters, so reducing bitrate is not always as effective as reducing packet retransmissions. Second, for many DNNs, the impact of missing a video frame depends not only on itself, but also on which other frames have been received or lost. We argue that some changes must be made in the *transport layer*, to determine whether to resend a packet based on the packet's impact on DNN's inference *dependent on* which packets have been received. While much research is needed towards an optimal design of DNN-driven transport layer, we believe that we have taken the first step in reducing streaming delay while maintaining a high inference accuracy.

1 INTRODUCTION

Video analytics (VA) applications and cheap video sensors are widely used. Videos collected by sensors are transmitted to cloud servers to run DNN-based inference. They can be used to detect vehicles for traffic safety analysis, track human face for better online conference experience, automatically grade player performance for cloud gaming, or even assist making decisions on operations in remote surgery. However, the use of VA applications is bounded by how accurate its analysis is and how long it takes to return the analysis results. For example, to achieve ideal service quality in augmented reality (which requires video analytics in order to properly interact with the surrounding environment), the ideal delay requirement should be around 5 ms [1].

To balance high accuracy and low delay, current VA systems use a range of methods to aggressively compress or prune video frames, and only send the most relevant frames or pixels from a camera (sender) to an analytical cloud server (receiver) for the DNN inference. For instance, AWStream [2] opportunistically lowers the video quality based on a profile of accuracy and bandwidth usage. DDS [3] aggressively compresses background regions while encoding potential regions of interest in high quality. Reducto [4] uses an elaborate heuristic to identify and discard frames that do not likely contain information relevant to the VA application.

Effective as they appear to be, these techniques are built on two assumptions: (1) without increasing available bandwidth, reducing delay means sending fewer bits, and (2) the impact of each video frame on DNN inference can be precisely determined before transmission. These assumptions work well for traditional TCP-based video streaming (*e.g.*, YouTube, Netflix), where lowering bitrate does effectively reduce transmission delay and the impact of missing a frame on quality (*e.g.*, SSIM) can be precisely computed by the sender. However, both assumptions should be revisited in real-time VA applications. First, a burst of packet losses can cause multiple rounds of packet retransmissions, so streaming delay will be dominated by retransmission delay and cannot be easily reduced by lowering video bitrate. Second, when a packet is lost, if its information relevant to DNN inference is also contained in some other received packets, the negative impact of not resending the lost packet will be much lower than otherwise (See Figure 2 for a concrete example). Thus, a packet's impact on the DNN inference is *dependent* on what packets have been received.

Instead, we believe that a new *transport-layer* solution, which selects which packets to resend based on *both* each packet's impact on DNN inference *and* which packets have been received, is pressingly needed. We refer to "transport layer" broadly as anything handling packet-level retransmission, including WebRTC [5] and QUIC [6] which manage packet transmission in user space. Existing transport-layer schemes, however, do not take into account the impact of packet losses on DNN inference. TCP retransmits every lost packet while UDP never retransmits. While forward-error correction (FEC) [7] can reduce packet retransmissions, it still treats all packets as equally important, so loss resilience is achieved only at the cost of sending more redundant bits. Recent work adds richer transport-layer semantics, but still assumes that the importance of each packet can be pre-determined regardless of the analytical DNN or what other packets have arrived. Bounded-loss transport [8], for instance, delivers a pre-defined fraction of packets, while selective retransmission [9, 10] in video streaming protects frames based on pre-defined impact.

This short paper takes the first step towards a DNN-aware transport scheme for real-time video analytics. Our design, `T4V`, determines whether to retransmit a packet based on the packet's *incremental impact*—how much impact the packet has on DNN inference *given* which packets have been received. A naive way to calculate each packet's incremental impact uses the output difference between DNN inference with and without the packet, but the compute cost would be prohibitive. `T4V` takes a more pragmatic stance to estimate the impact of each packet. Intuitively, a packet has a high impact if it belongs to a video frame whose pixels differ greatly from other received frames, or whose saliency (derivation of DNN's output with respect to the input pixels) is large (§3).

Thankfully, there are well-studied techniques to efficiently estimate frame differences [4] and pixel-level saliency [11], though

their effectiveness in T4V remains an open question. In this context, T4V's contribution is to demonstrate that a simple transport layer solution, based only on frame differences and saliency, can substantially improve real-time VA applications. Using a simulation on a typical VA application, T4V outperforms other transport layer solutions in terms of low transmission delay and high accuracy. Compared to TCP, T4V achieves similar accuracy but reduces packet retransmissions by more than 30%. Compared to UDP, T4V greatly reduces inaccuracy from 11% to 2%, at only a marginal delay inflation.

2 MOTIVATION AND BACKGROUND

Real-time VA applications need to stream video frames from a camera (sender) to a powerful cloud server (receiver) in order to run compute-intensive DNNs in real time [3]. This gives rise to the need to achieve high analytic accuracy while reducing the communication delay between the sender and the receiver.

Lowering bitrate is insufficient: Many recent VA systems have shown that, by compressing the video pixels/frames that have low relevance to the VA task, these applications can significantly reduce communication delay without hurting analytical accuracy. For instance, AWStream [2] opportunistically adjusts the video encoding parameters based on the incoming video content and the available bandwidth, achieving sub-second latency with only 2-6% accuracy drop; DDS [3] aggressively compresses video background while keeping high quality objects for object detection, reducing the communication delay by up to 59% for object detection applications without hurting the accuracy; Reducto [4] uses an elaborate heuristic to discard frames that do not likely contain information relevant to VA inference, reducing 51–97% of communication delay while consistently meets the desired accuracy.

These application-layer schemes are effectively under the assumption that reducing communication delay always requires sending fewer bits (lower bitrate). However, in real-time videos, it is known that lowering bitrate is less effective at reducing communication delay than minimizing packet retransmissions or enhancing robustness to packet losses [12]. When some packets are delayed or dropped (*e.g.*, due to bad wireless signals or transient congestion), resending them causes extra communication delay on every affected video frame. Figure 1 shows a real example where we run WebRTC (a widely used real-time video system) on a typical bandwidth trace. When bandwidth drops, the sending rate will temporarily exceed available bandwidth, causing congestion (queuing) and even packet drops, and increased frame delay to resend or wait for the delayed packets. Note that WebRTC uses Google Congestion Control (GCC) [13], which is known to be quite conservative at ramping up sending rate and aggressive at lowing sending rate, but packet losses are still hard to avoid. These observations corroborate with other measurements [14, 15].

These schemes also assume that the impact of each frame on DNN inference can be precisely pre-determined before being sent out. For instance, if receiving either of two frames and feeding it through the DNN yield similar output as feeding both frames as input, then whether to deliver each frame reliably will depend on whether the other frame is received. Intuitively, if a lost frame's information relevant to DNN inference is contained in some other received frames, the importance of the lost frame will be lower.

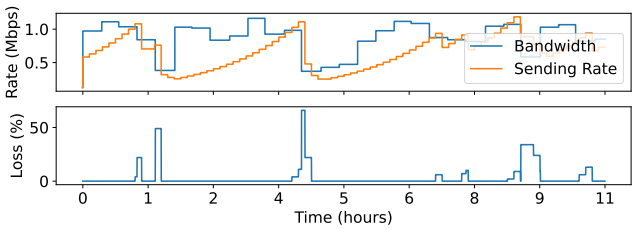

Figure 1: *In real-time videos, bandwidth drops can cause transient packet losses and increased frame-level delay.*

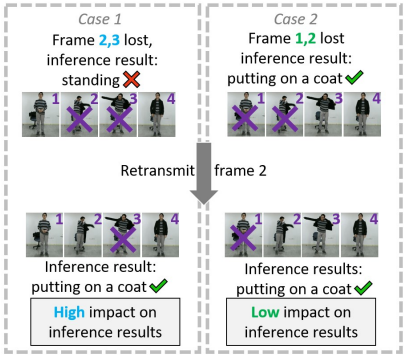

Figure 2: *Retransmiting frame 2 has high incremental impact when frame 2 and 3 are lost, but has low incremental impact when frame 1 and 2 are lost.*

Figure 2 shows a concrete example, in which whether a frame should be reliably delivered depends on which of other frames have already been received. Frame 2 and 3 are important for the DNN to correctly recognize the human action (putting on a coat). In Case 1, frame 2 and 3 are lost and the inference result is wrong because the information in 1 and 4 is not relevant enough for recognizing the action. Frame 2 has a great impact on the inference results as it provides relevant information. In case 2, however, frame 1 and 2 are lost, but frame 3 is received which provides relevant information such that the inference result is correct. The receipt of 3 means frame 2 has a low impact on the inference results as 3 can substitute 2 for recognising putting on a coat.

Current transport-layer schemes are ill-suited: Naturally, the transport-layer logic to decide packet retransmission should be aware of each frame's importance to DNN inference given the received frames. However, current transport layer schemes do not take into account the impact of a frame's loss on the DNN inference. TCP retransmits must reliably deliver all frames, so it has a high accuracy, but it also retransmits packet losses on frames that have little relevance to the analytic results (*e.g.*, frames that contain a person standing). UDP, on the other hand, never retransmits lost frames, but it may lose frames that are crucial to the analytic results (*e.g.*, frames that contain a person putting on a coat), resulting in low accuracy. Forward-error correction (FEC) [7] encodes extra information to recover the data of lost frames. However, the maximum number of frames that FEC can recover need to be set before the start of transmission. Thus, when the network condition is poor and the number of lost frames exceeds the pre-set maximum value, all lost frames are no longer recoverable, resulting in low accuracy.

Recent work adds richer transport-layer semantics, but still assumes that the importance of each packet can be pre-determined.

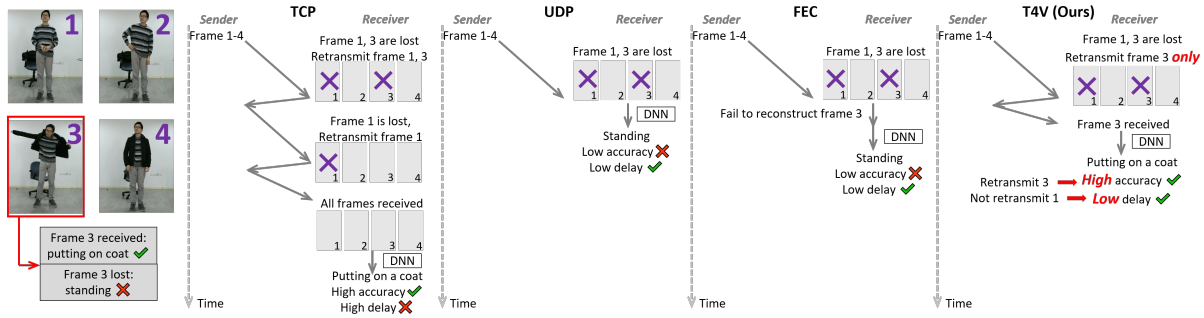

Figure 3: *Comparison of* T4V *with TCP, UDP and FEC. By only retransmitting frame 3,* T4V *achieves high accuracy while maintaining low delay.*

Bounded-loss transport [8] stops retransmitting frames only after a pre-determined number of frames are received. Selective retransmission [9] selects a subset of frames that are crucial for decoding the video and only retransmits this subset of frames. However, even if the video is successfully decoded, some frames that are crucial for DNN inference (*e.g.*, frames that contain a person putting on a coat) may still be lost and result in low accuracy. Quality-aware retransmission [10] works well in traditional video streaming scenarios. However, in real-time VA, the sender usually does not have enough compute to run the compute-intensive server-side DNN and estimate analytic accuracy As a result, this approach is not applicable to real-time VA applications.

In short, existing schemes in both the application layer and the transport layer are ill-suited for our goal. On one hand, current application-layer schemes focus on compressing the video and cannot handle retransmission-induced delay, when the network condition deteriorates. On the other hand, transport-layer schemes are agnostic to DNN inference.

3 KEY IDEAS OF T4V

We have seen that the packet retransmission decisions should be made dynamically based on what packets have been received at this point and the incremental impact of retransmitting a packet. Naturally, this goal cannot be achieved without a transport-layer scheme that manages packet-level transmission in real time. This section presents a simple-yet-effective design, called T4V (**T**ransport Scheme **for V**ideo Analytics).

To make this section more readable, without loss of generality, we assume that each frame has one packet, so losing the packet is equivalent to losing the frame. In practice, each frame might span one or more packets, which must arrive together to obtain the incremental impact of the frame.

Illustrative example: By estimating frames' incremental impact based on received frames, T4V only retransmits the important frames for inference. We take an example of action recognition to highlight the advantages of T4V over traditional transport-layer schemes. Figure 3 demonstrates 4 frames of a streamed video. It contains a person and the goal is to detect the person was putting on a coat. Frame 3 is important to inference as it contains the most relevant information about putting on a coat. We assume that in the first transmission, frame 1 and 3 are lost. TCP will retransmit 1 and 3 for multiple times until both are received, getting a correct inference result at the cost of high delay. UDP retransmits nothing, so the delay is small. But frame 3 is lost and the DNN mistakenly recognizes that the person was standing. FEC cannot reconstruct frame 3 even with the extra coded data because of exceeding the preset maximum recovery numbers. So the inference result is still wrong. Our design, T4V, only retransmits frame 3 leveraging incremental impact, so it gets a correct result with a trivial increase in delay. Besides baselines shown in Figure 3, DNN-aware T4V outperforms DNN-agnostic selective retransmission. In Figure 3, H.264-based selective retransmission will only retranmit frame 1, as H.264 usually picks the first and last frame to be I-frames for such a short video. The delay increase is as small as that of T4V, but the inference result is wrong because of the lost frame 3.

Estimation of incremental impact: To make retransmission decisions, the key of T4V is how to dynamically estimating the *incremental impact* of each packet—given the received packets, how much would adding the packet change the DNN output. However, estimating this impact can be computationally expensive. A naive way to calculate each packet's incremental impact would use the DNN output difference between DNN outputs with and without the packet, and the compute cost would be prohibitive.

T4V estimates a packet's incremental impact based on *frame differences* and *saliency*. We define them as follows (whose intuition will be explained shortly). The *frame difference* between the i^{th} frame and the j^{th} frame is $d(i,j) = \|F_i - F_j\|_1$, where F_i, F_j are the RGB-matrices of the i^{th} frame and j^{th} frame, $\|\cdot\|_1$ is l_1-norm. The *saliency* of the i^{th} frame is calculated as $sal(i) = \left\| \frac{\partial \mathbf{D}(F_i)}{\partial F_i} \right\|_2$, where $\mathbf{D}$ performs DNN inference on F_i and returns the confidence score output and $\|\cdot\|_2$ is the l^2-norm (*i.e.*, the gradient of the confidence scores of the inference results, accumulating over each pixel).

Intuitively, a frame tends to be important, if it has a significant difference to even the most similar frame among those that have been received. This aligns with previous work [4, 16]. We also observe that computer vision community widely uses saliency [17, 18] to describe frame importance. Intuitively, when a frame has very high saliency (based on the derivation of DNN's output with respect to the input pixels), this frame also tends to be important to the inference results.

As will be elaborated in (§5), T4V leaves as an open question how to efficiently estimate the packets' impact (frame difference and saliency) on inference. Thankfully, there are well-studied techniques that efficiently compute frame differences (based on low-level video features) [4], as well as recent proposals that predict pixel-level saliency using lightweight pre-trained DNN model [11].

Workflow: The workflow of T4V consists of four logical steps:

- *Encoding and packetization:* The sender compresses a video stream to a sequence of frames using a standard video codec like H.264 [19], and estimates the difference among nearby frames as well as the saliency of each frame. The frame difference and saliency values (no more than 63 bits, see §4 for details) will be stored in a custom header field of each packet, which is a negligible overhead per packet. This ensures that even some packets are dropped, the incremental impact of the dropped packets can be reconstructed by the receiver (cloud analytical server).
- *Video transmission:* The packets are then sent out over UDP. If the receiver does not receive a packet before a user-specified deadline or timeout, the packet is lost.
- *Packet retransmission:* If the i^{th} frame is lost, the receiver determines whether it should be retransmitted by

$$\text{Retransmit } i^{th} \text{ frame } \textbf{iff } sal(i) > T_s \textbf{ and } \min_{j \in \{\text{Received frames}\}} d(i,j) > T_d$$

  where T_s and T_d are tunable thresholds (by default, our experiment uses $T_s = 0.01, T_d = 0.001$). T_s enables T4V to retransmit only the packets that have great impact on the inference results, thus reducing delay but still improving accuracy. Optimal tuning of these thresholds is part of our future work, and we envision a carefully-designed heuristic similar to those for setting other thresholds in video analytics systems (*e.g.*, [4]).
- *Video padding:* As the inference results of the DNN might be affected by the number of the frames of the DNN input, we will pad the lost frames by the closest frames (*i.e.*, has the minimum frame difference) that have been already received.

Limitations of T4V: This simple design of T4V is not without weaknesses. For instance, it cannot handle the loss of all the packets from a moment on, as it does not know if there will be incoming frames. Similarly, we have not considered the implication of missing a frame on decoding of subsequent frames. This is reasonable if the sender encode frames using a reliably delivered frame as a reference (like in [20]), though doing so might have bitrate overhead.

4 CASE STUDY: ACTION RECOGNITION

As a case study, we use action recognition as our target application. Action recognition is a widely-used VA task (*e.g.*, using gesture to control smart home devices [21]), which takes a video clip as the input and outputs what human action is displayed in the video. For example, if a person is detected to be putting on a coat, the current temperature may be too low and the air conditioning system can start to heat up the room.

To obtain the delay of T4V and the baselines, we create a simple network simulator. The streaming delay of each packet is calculated as the size of transmitted packets divided by the bandwidth (default, 200Kbps) In this simulator, each frame is sent in one packet (1.5KB), which is consistent with the average frame size in low quality video (*e.g.*, 360p). Each frame is dropped with a random probability of 30%. The thresholds in T4V are: $T_s = 0.01$ and $T_d = 0.001$. To ensure statistical confidence, we run 100 rounds of independent tests to calculate the average results on all the baselines.

For the dataset, we randomly sample 12 video clips from Kinetics-400 [22] dataset, a well-known dataset for action recognition. We get 32 frames from each video clip. To ensure that the server can recover the saliencies and differences of all frames from any received frames, each frame has to be packed with 528 bits: 32 for whether the saliency of each frame is greater than the threshold T_u, 496 bits for whether the difference between any two out of the 32 frames is greater than the threshold T_d. Most of these bits are 0, so these bits can be very efficiently compressed.

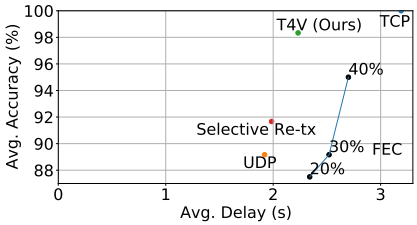

Figure 4: *Average inference accuracy versus average delay of action recognition simulation with a frame loss rate of 30%.*

For the server-side analytical model, we take I3D [23], pre-trained on Resnet-50 [24] by GluonCV [25], as the server side DNN. The server uses one Nvidia GeForce RTX 3080 for inference.

For the baselines, we use TCP, UDP, FEC and H.264-based selective retransmission. TCP works as the groundtruth of analytic accuracy. FEC's [7] maximum tolerance of loss rate is set to be a range centered at the loss rate. Selective retransmission [9] is based on H.264 codecs. Bounded-loss transport [8] is not selected because it works like selective retransmission whose selection is based on how many packets have been received. Quality-aware retransmission [10] is not selected as it does not fit into real-time VA applications.

The simulation results, as shown in Figure 4, prove the effectiveness of T4V's estimation of incremental impact. Compared to TCP, T4V achieves similar accuracy but reduces packet retransmissions by more than 30% and thus significantly reduces the delay of real-time VA. Compared to UDP, T4V greatly reduces inaccuracy from 11% to 2%, at only a marginal delay inflation (15%). Compared to H.264-based selective retransmission, T4V reduces inaccuracy from 8% to 2% with only 10% delay increase. Compared to FEC, T4V achieves higher accuracy by 3-13% at a reduced delay (by 8-19%).

5 OPEN QUESTIONS

In designing T4V, we have taken the first step in demonstrating the potential of a DNN-aware transport for real-time video analytics. Yet, to make it truly practical, a few key questions remain open. Here, we conclude the paper by highlighting two open questions.

Saliency estimation: Estimates of pixel-/frame-level saliency play a critical role not only in T4V but other recent systems (*e.g.*, [11, 18]) for various purposes. Though precise saliency values requires running forwardprop and backwardprop on a large DNN, thankfully these systems suggest that the saliency values can be reasonably approximated by training cheap predictors (*e.g.*, a shallow NN). We believe that future work to study these saliency estimators will shed light on effectiveness of existing estimators and how to improve them for DNN-aware packet transmission.

Faster retransmission decisions: For any transport-layer scheme to succeed, it is vital to keep control logic separate from the fast path of packet delivery [26]. This is particularly true in T4V whose retransmission decisions require non-trivial computation. One candidate solution is to move some compute to sensors to leverage their local compute power (like in [4, 16]). Another solution is to

pipeline packet retransmission with DNN inference on those received frames. This is apt especially when DNN inference runs on a similar timescale as packet retransmission, thus hiding the retransmission delay.

REFERENCES

[1] 2019. GSMA | Cloud AR/VR Whitepaper - Future Networks. https://www.gsma.com/futurenetworks/wiki/cloud-ar-vr-whitepaper/#_ftn9. (Accessed on 06/17/2022).

[2] Ben Zhang, Xin Jin, Sylvia Ratnasamy, John Wawrzynek, and Edward A. Lee. 2018. AWStream: Adaptive Wide-Area Streaming Analytics. In *Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication* (Budapest, Hungary) *(SIGCOMM '18)*. Association for Computing Machinery, New York, NY, USA, 236–252. https://doi.org/10.1145/3230543.3230554

[3] Kuntai Du, Ahsan Pervaiz, Xin Yuan, Aakanksha Chowdhery, Qizheng Zhang, Henry Hoffmann, and Junchen Jiang. 2020. Server-Driven Video Streaming for Deep Learning Inference. In *Proceedings of the Annual Conference of the ACM Special Interest Group on Data Communication on the Applications, Technologies, Architectures, and Protocols for Computer Communication* (Virtual Event, USA) *(SIGCOMM '20)*. Association for Computing Machinery, New York, NY, USA, 557–570. https://doi.org/10.1145/3387514.3405887

[4] Yuanqi Li, Arthi Padmanabhan, Pengzhan Zhao, Yufei Wang, Guoqing Harry Xu, and Ravi Netravali. 2020. Reducto: On-Camera Filtering for Resource-Efficient Real-Time Video Analytics. In *Proceedings of the Annual Conference of the ACM Special Interest Group on Data Communication on the Applications, Technologies, Architectures, and Protocols for Computer Communication* (Virtual Event, USA) *(SIGCOMM '20)*. Association for Computing Machinery, New York, NY, USA, 359–376. https://doi.org/10.1145/3387514.3405874

[5] Niklas Blum, Serge Lachapelle, and Harald Alvestrand. 2021. WebRTC - Realtime Communication for the Open Web Platform: What Was Once a Way to Bring Audio and Video to the Web Has Expanded into More Use Cases We Could Ever Imagine. *Queue* 19, 1 (feb 2021), 77–93. https://doi.org/10.1145/3454122.3457587

[6] Adam Langley, Alistair Riddoch, Alyssa Wilk, Antonio Vicente, Charles Krasic, Dan Zhang, Fan Yang, Fedor Kouranov, Ian Swett, Janardhan Iyengar, Jeff Bailey, Jeremy Dorfman, Jim Roskind, Joanna Kulik, Patrik Westin, Raman Tenneti, Robbie Shade, Ryan Hamilton, Victor Vasiliev, Wan-Teh Chang, and Zhongyi Shi. 2017. The QUIC Transport Protocol: Design and Internet-Scale Deployment. In *Proceedings of the Conference of the ACM Special Interest Group on Data Communication* (Los Angeles, CA, USA) *(SIGCOMM '17)*. Association for Computing Machinery, New York, NY, USA, 183–196. https://doi.org/10.1145/3098822.3098842

[7] Frank Chang, Kiyoshi Onohara, and Takashi Mizuochi. 2010. Forward error correction for 100 G transport networks. *IEEE Communications Magazine* 48, 3 (2010), S48–S55. https://doi.org/10.1109/MCOM.2010.5434378

[8] Jiacheng Xia, Gaoxiong Zeng, Junxue Zhang, Weiyan Wang, Wei Bai, Junchen Jiang, and Kai Chen. 2019. Rethinking Transport Layer Design for Distributed Machine Learning. In *Proceedings of the 3rd Asia-Pacific Workshop on Networking 2019* (Beijing, China) *(APNet '19)*. Association for Computing Machinery, New York, NY, USA, 22–28. https://doi.org/10.1145/3343180.3343186

[9] Nick Feamster and Hari Balakrishnan. 2002. Packet Loss Recovery for Streaming Video. In *In 12th International Packet Video Workshop*.

[10] Mirko Palmer, Malte Appel, Kevin Spiteri, Balakrishnan Chandrasekaran, Anja Feldmann, and Ramesh K. Sitaraman. 2021. *VOXEL: Cross-Layer Optimization for Video Streaming with Imperfect Transmission*. Association for Computing Machinery, New York, NY, USA, 359–374. https://doi.org/10.1145/3485983.3494864

[11] Kuntai Du, Qizheng Zhang, Anton Arapin, Haodong Wang, Zhengxu Xia, and Junchen Jiang. 2022. AccMPEG: Optimizing Video Encoding for Accurate Video Analytics. In *Proceedings of Machine Learning and Systems*, D. Marculescu, Y. Chi, and C. Wu (Eds.), Vol. 4. 450–466. https://proceedings.mlsys.org/paper/2022/file/98f13708210194c475687be6106a3b84-Paper.pdf

[12] 2021. Working latency — the next QoE frontier. https://blog.apnic.net/2021/12/02/working-latency-the-next-qoe-frontier/.

[13] Gaetano Carlucci, Luca De Cicco, Stefan Holmer, and Saverio Mascolo. 2016. Analysis and design of the google congestion control for web real-time communication (WebRTC). In *Proceedings of the 7th International Conference on Multimedia Systems*. 1–12.

[14] 2021. Focusing on latency, not throughput, to provide a better internet experience and network quality. https://www.iab.org/wp-content/IAB-uploads/2021/09/Nokia-IAB-Measuring-Network-Quality-Improving-and-focusing-on-latency-.pdf.

[15] 2021. A Single Common Metric to Characterize Varying Packet Delay. https://www.iab.org/wp-content/IAB-uploads/2021/09/single-delay-metric.pdf.

[16] Tiffany Yu-Han Chen, Lenin Ravindranath, Shuo Deng, Paramvir Bahl, and Hari Balakrishnan. 2015. Glimpse: Continuous, real-time object recognition on mobile devices. In *Proceedings of the 13th ACM Conference on Embedded Networked Sensor Systems*. 155–168.

[17] Timor Kadir and Michael Brady. 2001. Saliency, scale and image description. *International Journal of Computer Vision* 45, 2 (2001), 83–105.

[18] Qizheng Zhang, Kuntai Du, Neil Agarwal, Ravi Netravali, and Junchen Jiang. 2022. Understanding the potential of server-driven edge video analytics. In *Proceedings of the 23rd Annual International Workshop on Mobile Computing Systems and Applications*. 8–14.

[19] Iain E. Richardson. 2010. *The H.264 Advanced Video Compression Standard* (2nd ed.). Wiley Publishing.

[20] Sadjad Fouladi, John Emmons, Emre Orbay, Catherine Wu, Riad S Wahby, and Keith Winstein. 2018. Salsify:{Low-Latency} Network Video through Tighter Integration between a Video Codec and a Transport Protocol. In *15th USENIX Symposium on Networked Systems Design and Implementation (NSDI 18)*. 267–282.

[21] Fariz Alemuda and Fuchun Joseph Lin. 2017. Gesture-based control in a smart home environment. In *2017 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)*. IEEE, 784–791.

[22] Will Kay, Joao Carreira, Karen Simonyan, Brian Zhang, Chloe Hillier, Sudheendra Vijayanarasimhan, Fabio Viola, Tim Green, Trevor Back, Paul Natsev, Mustafa Suleyman, and Andrew Zisserman. 2017. The Kinetics Human Action Video Dataset. arXiv:1705.06950 [cs.CV]

[23] Joao Carreira and Andrew Zisserman. 2017. Quo Vadis, Action Recognition? A New Model and the Kinetics Dataset. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[24] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep Residual Learning for Image Recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[25] Jian Guo, He He, Tong He, Leonard Lausen, Mu Li, Haibin Lin, Xingjian Shi, Chenguang Wang, Junyuan Xie, Sheng Zha, Aston Zhang, Hang Zhang, Zhi Zhang, Zhongyue Zhang, Shuai Zheng, and Yi Zhu. 2020. GluonCV and GluonNLP: Deep Learning in Computer Vision and Natural Language Processing. *Journal of Machine Learning Research* 21, 23 (2020), 1–7. http://jmlr.org/papers/v21/19-429.html

[26] Akshay Narayan, Frank Cangialosi, Prateesh Goyal, Srinivas Narayana, Mohammad Alizadeh, and Hari Balakrishnan. 2017. The case for moving congestion control out of the datapath. In *Proceedings of the 16th ACM Workshop on Hot Topics in Networks*. 101–107.