Dejavu: Enhancing Videoconferencing with Prior Knowledge

Pan Hu
Stanford University
panhu@stanford.edu

Rakesh Misra
Uhana Inc.
rakesh@uhana.io

Sachin Katti
Stanford University
skatti@cs.stanford.edu

ABSTRACT

Videoconferencing over the Internet routinely suffers from poor quality as videoconferencing systems, in order to guarantee interactive delays which is critical to user experience, are commonly designed to stream at conservative qualities in the face of variable bandwidths. In this paper, we present Dejavu, a system that enables existing videoconferencing systems to alleviate this problem. The key insight that powers Dejavu is that recurring videoconferencing sessions, e.g., in the same conference room or by the same person, have a lot of visual similarities that can be encoded based on the sender's historical videoconferencing sessions, and shared with the receiver in advance. Accordingly, Dejavu first learns an offline mapping between low-quality and high-quality versions of frames in the sender's past videoconferencing sessions, and then applies this mapping in real time at the receiver to convert the low-quality frames into high-quality frames. As a result, a videoconferencing system equipped with Dejavu can continue to stream at conservative qualities to guarantee interactive delays like today, but can now additionally *enhance* the video quality at the receiver. Our evaluation shows that Dejavu can provide a 1.3 dB increase in PSNR for the same bandwidth consumption, or equivalently save up to 30% in bandwidth to deliver the same PSNR.

CCS CONCEPTS

• **Networks** → **Mobile networks**; • **Information systems** → *Multimedia streaming*.

KEYWORDS

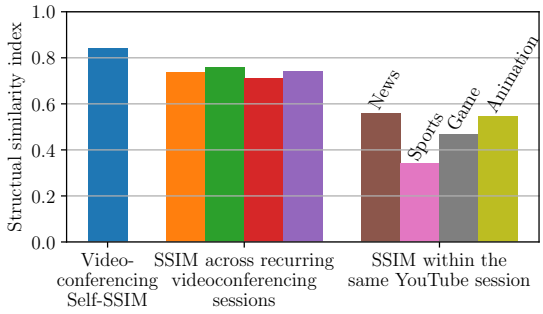
Videoconferencing; Mobile networks; Deep neural networks

ACM Reference Format:
Pan Hu, Rakesh Misra, and Sachin Katti. 2019. Dejavu: Enhancing Videoconferencing with Prior Knowledge. In *The 20th International Workshop on Mobile Computing Systems and Applications (HotMobile '19), February 27–28, 2019, Santa Cruz, CA, USA.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3301293.3302373

1 INTRODUCTION

Videoconferencing over the Internet routinely suffers from poor quality [16]. The reason is that videoconferencing systems need to deliver interactive end-to-end delays for the best user experience, so when network bandwidths are variable, in order to still guarantee interactive delays, the common approach is to stream at very conservative qualities. As a result, in networks where user bandwidths are dynamic, e.g., in mobile networks, the video quality suffers. Most of the work in addressing this problem has focused on designing more-efficient video encoders [6, 7] and better real-time transport algorithms [4, 16] for videoconferencing.

In this paper, we adopt a different approach. Our guiding insight is that recurring videoconferencing sessions, i.e., sessions that happen in the same meeting room and/or involve the same person, have a lot of similarities in visual content *across* sessions. For example, many meeting rooms use a fixed camera, so the objects in the field of view like conference table, whiteboard etc. are largely the same in all sessions using that camera. Similarly, personal videoconferencing using a mobile or web camera always has the face and features of the same person. Note that this property of videoconferencing streams does not hold in general for any live or on-demand



(a) Structural similarity (SSIM) index between different videos

(b) Sample frames from **different** videoconferencing sessions

(c) Sample frames from the **same** Youtube video stream

Figure 1: The key insight that powers Dejavu is that recurring videoconferencing sessions, e.g., in the same meeting room, are visually very similar. As (a) shows, the SSIM of frames *across* recurring videoconferencing sessions (0.70-0.75) is higher than the SSIM of even frames within the *same* video session involving news (0.56), sports (0.34), gaming (0.47) or animation (0.55). The sample frames in (b) and (c) illustrate this insight visually.

HotMobile '19, February 27–28, 2019, Santa Cruz, CA, USA

ACM ISBN 978-1-4503-6273-3/19/02…$15.00
https://doi.org/10.1145/3301293.3302373

video stream. For example, as we show in Figure 1, frames *across* multiple videoconferencing sessions in a meeting room using a fixed camera had a significantly higher structural similarity index (0.70-0.75) than even frames *within* the same live/on-demand video streaming session involving news (0.56), sports (0.34), gaming (0.47) or animation (0.55).

The above insight led us to the following question: if there is so much similarity in the visual content *across* sessions involving the same sender, could we learn a mapping between the low-quality and the high-quality versions for a sender based on her past sessions, and share it with the receiver *before* a live session starts? If we could, then we could use this mapping at the receiver *during* a live session to up-convert the video quality in real time. As a result, existing videoconferencing systems could deliver the best of both worlds: they could continue to deliver interactive delays since the streaming over the network will continue to happen at the same qualities as today[1], but now they could also deliver better qualities as the receiver will *enhance* the video quality in real time. In effect, the spare computing power at the receivers could be employed to compensate for the conservative quality choices at the sender[2].

In order to realize the above vision, we present Dejavu, a system that can augment existing videoconferencing systems and enable them to deliver better video qualities while continuing to deliver the interactive delays that they do today. Dejavu shows that it is possible to learn a quality-enhancing *model* or mapping based on a sender's historical sessions that performs well in enhancing the video quality of future unseen sessions. Our initial evaluation shows that Dejavu can help videoconferencing systems improve their end-to-end PSNR by more than 1.3 dB while consuming the same bandwidth as today, or alternatively enable them to reduce their bandwidth consumption by 30% (thereby potentially leading to lower delays) while delivering the same PSNR as today.

2 DESIGN

In this section, we describe how Dejavu has been designed to realize the above vision. Dejavu operates in two stages, as shown in Figure 2.

In the offline stage, Dejavu uses videos from a sender's past videoconferencing sessions to: (i) generate training data by re-encoding the higher-quality frames at each resolution, e.g., 800 kbps at 540p, to lower qualities at the same resolution, e.g., 500 kbps at 540p, using the same encoding pipeline as the videoconferencing system, and (ii) train a deep neural network model using Dejavu's learning engine that learns the mapping between the lower-quality frames of a given resolution and their corresponding higher-quality frames, e.g., model to convert frames at 500 kbps at 540p to frames at 800 kbps at 540p. These models are shared with the Dejavu module at the receiver before the start of the next videoconferencing session.

[1]In reality, the additional processing at the receiver will add some delay, so for this to work, a requirement is that the additional delay has to be very minimal.

[2]This idea of using spare receiver compute resources to offset the loss in sending quality due to limited network bandwidths has been explored in a recent paper [18] but in the context of *on-demand* video streaming; as we describe in Section 6, learning an offline mapping for *on-demand* videos where the entire video content is known in advance is very different than that for *live* videos where the video content whose quality has to be enhanced is unseen and therefore not known in advance.

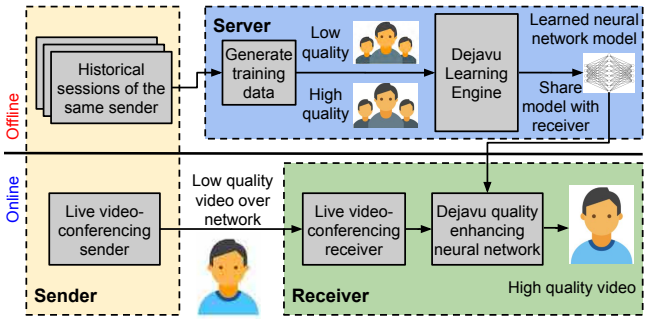


Figure 2: The Dejavu system consists of two components: an offline component that learns a quality-enhancing deep neural network, and an online component that enhances the quality of live videoconferencing at the receiver in real time.

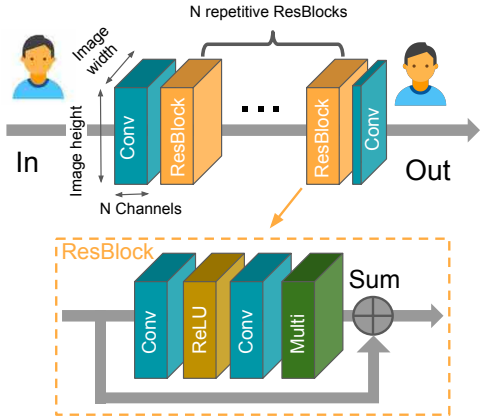


Figure 3: Dejavu's quality-enhancing neural network consists of convolutional layers and residual blocks [11]. It converts a low-quality image into a high-quality image at the same resolution.

In the online stage, the Dejavu module at the receiver applies the pre-trained neural network models on the conventional videoconferencing receiver output to up-convert the quality, e.g., from 500 kbps at 540p to 800 kbps at 540p, in real time.

2.1 Architecture of the quality-enhancing neural network

At the heart of Dejavu is the quality-enhancing neural network whose goal is to learn a mapping between the low-quality input frames and the corresponding high-quality output frames. We revised the neural network architecture proposed by EDSR [14] by removing bi-cubic upsampling layers so the output has the same dimension as input, which is also shown in Figure 3.

Dejavu's deep neural network consists of several repetitive ResNet [11] blocks sandwiched between two convolutional layers. Each ResBlock consists of a convolutional layer, a rectified linear unit (ReLU) layer, another convolutional layer and a multiply layer. There is a

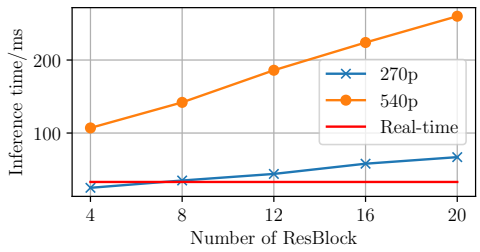


Figure 4: Comparison of the inference speeds of the quality-enhancing neural networks with different number of ResBlocks for different input resolutions.

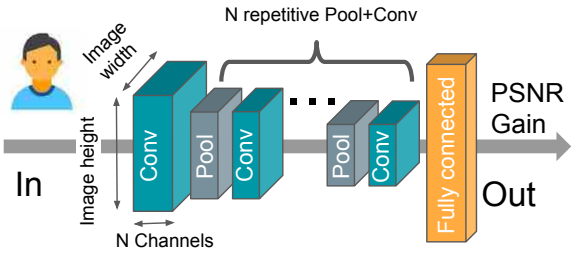


Figure 5: Patch-scoring neural network consists of shallow layers of max-pooling and convolutional blocks. It outputs a single floating number representing the PSNR gain if the quality-enhancing network were to be applied on the input patch.

Figure 6: Samples images from the training dataset for Dejavu: five colleagues participating in mock interviews in the same conference room.

skip connection from input to output of the ResBlock that helps in avoiding the vanishing gradient problem [11].

Instead of designing a neural network that works on the entire input image, we instead design it to work on smaller patches of the image. For example, instead of a neural network that works on a single 960×540 RGB image which would require 1.56M parameters, we instead design a neural network that can work on 144 80×45 patches; each neural network then has only 10.8k parameters. The reduced input size allows us to use deeper neural networks as well as more mini-batches for stable training.

2.2 Inference speed

A key challenge that Dejavu has to solve is in speeding up the inference of the deep neural network so it can be scored in real time, even with less powerful GPUs.

Figure 4 shows a comparison of inference speeds as a function of the number of ResBlocks at different input resolutions, using a high-end, NVIDIA Tesla P100 from Google Cloud. Assuming the video frame rate is 30 fps, we find that only very shallow networks can meet the real-time deadline of 33 ms, shown in red in the figure.

Dejavu applies two techniques to speed up inference:

Train/infer on Y (luminance) channel only: It is well known that human eyes are most sensitive to the luminance component of an image, and less sensitive to its colors. Therefore, we convert an RGB image into its equivalent YCbCr representation, where Y is the luminance component, and Cb and Cr are the two chrominance components. We found that by training/inferring a neural network on only the Y channel, while keeping Cb and Cr channels the same, we can still achieve significant gains in visual quality while taking only one third of the time for inference (the cost of converting RGB into YCbCr is minimal).

Patch scoring network: Dejavu's quality-enhancing neural network does not achieve the same improvement in visual quality on all patches. It is possible to improve the quality more on patches with a lot of edges or complex details, rather than patches with, for example, a white wall. So instead of scoring the quality-enhancing neural network on all patches, Dejavu has a patch-scoring network that predicts possible gains in quality, measured in peak SNR, as shown in Figure 5. During the inference process, the patch-scoring network ranks the patches according to their predicted gains in quality. Dejavu selects the top k patches according to the available compute resources so as to meet the real-time requirement.

3 IMPLEMENTATION

We implement the neural network with TensorFlow [1] and Python 3. All the training and inference are done in Google Cloud for now. The trained neural network model is typically less than 870kB given 32 residual blocks and 32 features per each block, which can be sent to the receiver easily.

Video dataset. Since there is no widely-acknowledged dataset for videoconferencing, we created our own videoconferencing dataset by requesting five people to do mock interviews with us in our meeting room on the same day. Figure 6 shows some sample images from this data set. The camera was pointed towards a fixed direction to simulate conference room settings. We used a smartphone with mini-tripod to capture 1080p@60fps video.

Training dataset: Each video is about three minutes in length. All the videos were converted into VP9 format [1]. In videoconferencing, the video stream can be encoded at any bitrate ranging from between tens of kbps to several Mbps. We chose 7 representative bitrates in our experiment, including: 100, 200, 300, 500, 800, 1000, 2000 kbps. We compressed each video into different bitrate levels using FFmpeg, and used OpenCV [2] to extract compressed frames for training. To reduce the size of dataset, we reduced the frame rate from 60 fps to 2 fps. So the total number of frames is about 5 videos * 180 seconds * 2 fps * 8 levels = 12600. We then converted each frame into lossless png format to facilitate data loading.

The videos used to calculated similarity were downloaded from YouTube, including a CNN clip (news), a 2018 FIFA clip (sports), a

[1]We chose VP9 over H265 as VP9 is free and benchmarks show that VP9 has similar performance [9].

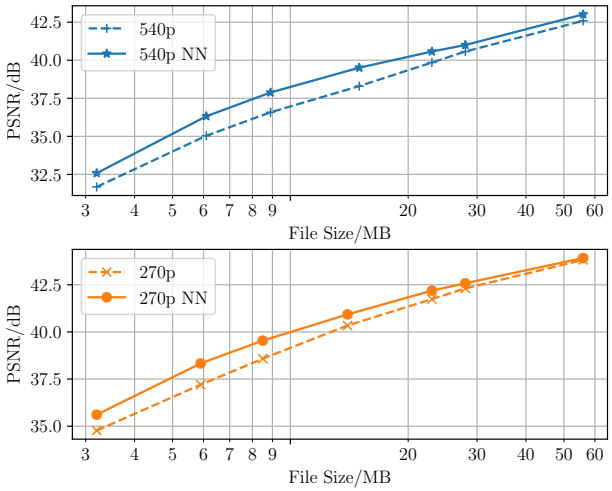


Figure 7: PSNR gain of Dejavu for inputs at two resolutions: 540p (blue) and 270p (orange). Each marker represents an encoding bitrate ranging from 100 kbps to 2000 kbps.

Dota v2 clip (gaming) and a Family Guys clip (animation), all of which are top-ranking 1080p HD videos.

4 EVALUATION

In this section we present some preliminary evaluation results for Dejavu. We aim to answer the following questions:

- What performance gains can Dejavu provide and how do we interpret the results?
- How does video similarity affect the performance of the quality-enhancing neural network?

We use PSNR [1] as the quality metric to quantify the performance gain of the quality-enhancing network. We train on four out of the five videos in the dataset and test on the fifth one. The results for 270p and 540p videoconferencing streams are shown in Figure 7, where x axis is the logarithm of the file size in MB and y axis is the PSNR in dB. Each marker indicates an encoding bitrate level in [100, 200, 300, 500, 800, 1000, 2000] kbps.

The file sizes grow monotonically as the encoding bitrates increase. As the figure shows, the quality-enhancing neural network is able to improve the PSNR by up to 1.3 dB for 540p and up to 1.1 dB for 270p videoconferencing streams at low to medium encoding bitrates. This gain is substantial as it is comparable to developing a new generation of video codec [8] which requires years of work. The gains diminish as the file sizes / encoding bitrates increase, as the room for improvement decreases.

Figure 8 shows sample pictures of the input/output of the neural network, and the corresponding ground truth. The PSNR and SSIM gains aside, we can see that the details are clearly refined in the output of the neural network with much less compression artifacts. The drawing on the white board is also much more readable. In Figure 9, we plot a differential error map that shows the boost in image quality due to Dejavu in different parts of the image [2]. The interesting observation is that most of the gain (shown in purple) occurs around edges.

We provide an alternative way to understand the performance gain by translating the result from Figure 7 into potential bandwidth savings, as shown in Figure 10. The figure shows that we can save more than 30% of the bandwidth to deliver video at the same quality as the original 540p@500kbps stream, while we can also achieve more than 25% bandwidth saving at 270p.

Another question we want to answer is: what is the performance gain if we train on a general video dataset and use the model to enhance videoconferencing sessions? The result is shown in Figure 11: the quality-enhancing neural network trained on a general video data set cannot bring significant PSNR gain (<0.1dB), and is significantly worse than the neural network trained on the videoconferencing dataset. The result is also in accordance with the similarity result shown in Figure 1 at the beginning of the paper, indicating that it is hard to improve the video quality if there is less similarity between the train and the test datasets.

5 FUTURE WORK

While the above results prove the potential of Dejavu in improving the visual quality of videoconferencing streams, a number of open items still need to be completed in order to make Dejavu ready for deploying in practical videoconferencing systems.

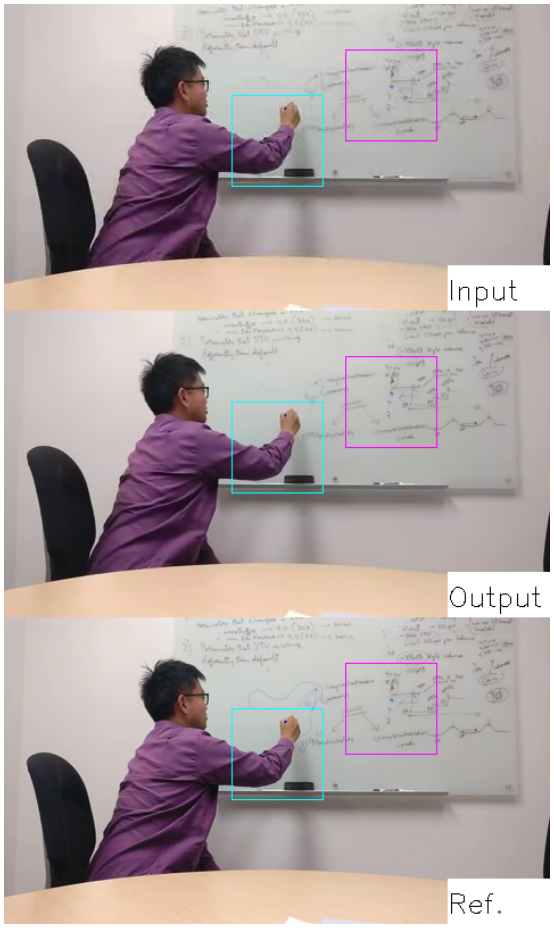
1. Evaluating the performance of Dejavu in the wild: The evaluation in this paper was performed for a simple, same-room different-person scenario. There is need for a larger-scale evaluation that collects data from more videoconferencing scenarios, including multiple persons in the same meeting room, same person but at different locations on different days, as well as scenarios that use front-facing camera on the go. Such data is necessary to better evaluate the real-world performance of Dejavu.

2. Integrating Dejavu into a videoconferencing application and measuring user experience: Our initial evaluation focused only on improving the video quality without taking into account the additional processing delay incurred at the receiver. However, optimizing delay and jitter are crucial for a good user experience with videoconferencing. Therefore, there might be a need to optimize the algorithm that selects video sending rates in videoconferencing frameworks like WebRTC in order to compensate for the additional processing at the receiver.

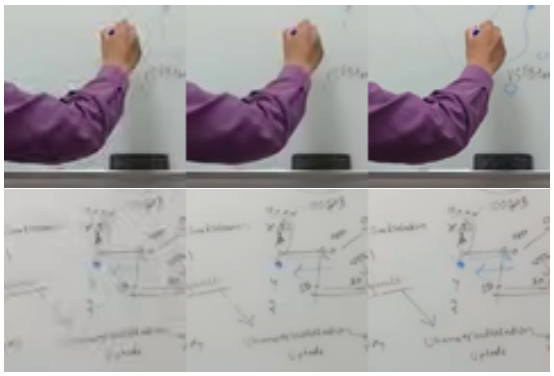
3. Supporting mobile / embedded devices with resource constraints: We used powerful GPUs in our current evaluation. However, not every videoconferencing device, especially mobile phones and embedded devices, can afford such a computational cost. Although sometimes it is possible to perform the computations in the cloud using a relay server if the receiver has a good downlink, relaying incurs additional delay as well as additional costs for the service provider. Therefore, there is a need to optimize Dejavu for devices with compute constraints. *Model compression* [10] and *knowledge distillation* [12] are two popular methods for reducing computation overhead: model compression compresses neural networks via pruning less important connections as well as quantizes

[1] $PSNR = 48.13 - 10 \times \log_{10}(MSE)\ dB$ for 8 bits per sample image where MSE is mean squared error between the input and output image.

[2] We clip the data range from [-79, 61] to [-20, 20] for better color contrast.



(a) Top: input to Dejavu (**36.17** dB PSNR, 0.9445 SSIM); middle: output of Dejavu (**37.53** dB PSNR, 0.9617 SSIM); bottom: reference (ground-truth) frame.



(b) Zoomed in versions of the blue and the purple boxes in (a). Left to right: input, output, reference (ground truth).

Figure 8: Illustration of Dejavu's quality enhancement

weights so we can use less bits per weight; knowledge distillation involves training a smaller "student" model that mimics the behavior of the larger "teacher" model.

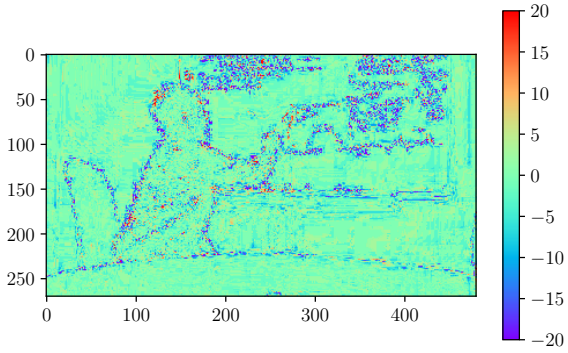


Figure 9: Error map showing how Dejavu improves quality over different areas of a frame, computed as a pixel-wise subtraction ($|I_{out} - I_{ref}| - |I_{in} - I_{ref}|$). Purple pixels indicate that Dejavu reduces the error at that pixel by 20 while red indicates Dejavu increases error by 20 (closer to purple means better performance). Notice that most of the improvement occurs around edges but red pixels could be very close to purple pixels.

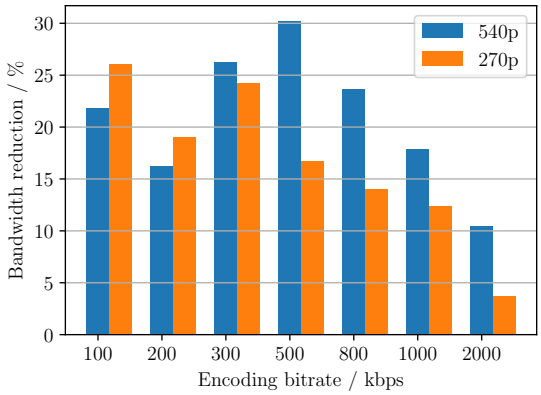


Figure 10: Equivalent bandwidth savings according to the PSNR gains of Dejavu for different encoding bitrates and input resolutions.

4. Exploiting inter-frame similarity: Dejavu currently processes every frame at the receiver using the same neural network; however there is an opportunity to design separate neural networks for key frames and predicted frames. Predicted frames are more common but carry less information, therefore it might be possible to enhance their quality using a smaller (shallower and less wide) neural network, whereas the current full-scale neural network could enhance only the key frames. As a result, the average run time of Dejavu at the receiver could be significantly reduced.

6 RELATED WORK

Dejavu's quality-enhancing neural network is built upon prior works in video super resolution [3, 15] and image super resolution [5, 13, 14]. Dejavu's novel contribution is to adapt these prior

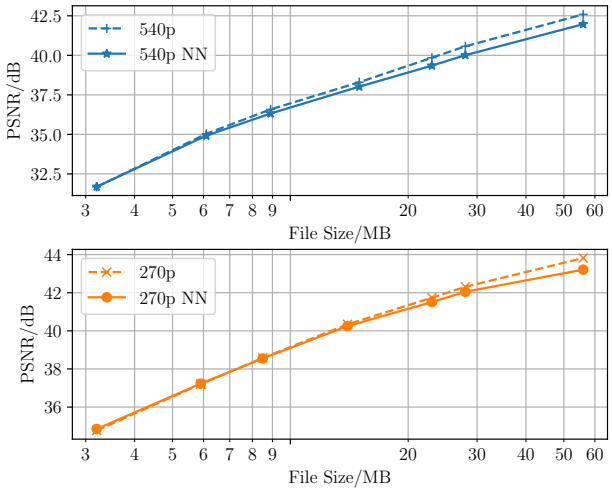


Figure 11: Comparison of PSNR gain when the quality-enhancing model is trained on general Youtube videos. The gains in PSNR are negligible (< 0.1 dB in most cases).

neural networks to exploit the unique properties of videoconferencing, and build a system that shows how such neural networks can be used in an end-to-end application like videoconferencing.

Dejavu shares similarities with NAS [17, 18] that explored a similar quality-enhancing problem but in the context of *on-demand video streaming*. Dejavu and NAS are similar in the sense that both aim to learn a mapping between the low-quality and the high-quality versions of video streams, and both aim to use the spare compute resources at the receiver to enhance the video quality at run time. However, Dejavu differs in two significant respects:

(i) While NAS needs to learn a *model that overfits* to this mapping for a *given and known* video stream, Dejavu needs to learn a *model that predicts* this mapping for *future unseen* video streams. This is because in on-demand video streaming, the video stream whose quality has to be enhanced is entirely known in advance, where as in live videoconferencing, the quality enhancement has to be performed on video streams that are unseen until run time. Dejavu is therefore uniquely designed to solve a learning problem that is significantly different than the one solved by NAS.

(ii) While NAS aims to *increase the resolution* of a video stream, Dejavu aims to *increase the effective encoding rate while retaining the same resolution*. This is based on the insight that given a bandwidth target, in videoconferencing, it is usually better to send at a higher resolution using lesser bits per pixel than at a lower resolution using more bits per pixel. Therefore, in the context of videoconferencing, it is almost always more useful to be able to increase the effective bits per pixel, unlike in traditional live or on-demand video streaming where the choice largely depends on the nature of content in the video stream.

There have been works on jointly controlling video codec and transport protocols [6] that aim to improve the quality of real-time streaming. While Dejavu addresses the same problem, it adopts a different approach based on taking advantage of the similarities across recurring videoconferencing sessions.

7 CONCLUSION

In this paper, we discussed the design and evaluation of Dejavu, a system that enhances videoconferencing quality by extracting prior knowledge from historical sessions. We believe that Dejavu has great potential to improve existing videoconferencing systems as initial evaluation shows up to 1.3dB PSNR gain, or equivalently 30% savings in bandwidth. Our ongoing work is focused on addressing the open items in Section 5 to make Dejavu ready for deployment in practical large-scale videoconferencing systems.

Acknowledgements: We would like to thank our reviewers and our shepherd Eric Rozner for their valuable reviews. We would also like to thank Tianshu Chu, Keith Winstein and Samuel Joseph for their feedback on early drafts of this paper, and Stanford Platform Lab for supporting us.

REFERENCES

[1] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, et al. Tensorflow: a system for large-scale machine learning. In *OSDI*, volume 16, pages 265–283, 2016.

[2] G. Bradski and A. Kaehler. Opencv. *Dr. Dobbs journal of software tools*, 3, 2000.

[3] J. Caballero, C. Ledig, A. P. Aitken, A. Acosta, J. Totz, Z. Wang, and W. Shi. Real-time video super-resolution with spatio-temporal networks and motion compensation. In *CVPR*, volume 1, page 7, 2017.

[4] G. Carlucci, L. De Cicco, S. Holmer, and S. Mascolo. Analysis and design of the google congestion control for web real-time communication (webrtc). In *Proceedings of the 7th International Conference on Multimedia Systems*, page 13. ACM, 2016.

[5] C. Dong, C. C. Loy, K. He, and X. Tang. Image super-resolution using deep convolutional networks. *IEEE transactions on pattern analysis and machine intelligence*, 38(2):295–307, 2016.

[6] S. Fouladi, J. Emmons, E. Orbay, C. Wu, R. S. Wahby, and K. Winstein. Salsify: Low-latency network video through tighter integration between a video codec and a transport protocol. In *15th USENIX Symposium on Networked Systems Design and Implementation (NSDI 18)*. USENIX, 2018.

[7] A. Grange and H. Alvestrand. A vp9 bitstream overview, draft-grange-vp9-bitstream-00, 2013.

[8] D. Grois, D. Marpe, A. Mulayoff, B. Itzhaky, and O. Hadar. Performance comparison of h. 265/mpeg-hevc, vp9, and h. 264/mpeg-avc encoders. In *Picture Coding Symposium (PCS), 2013*, pages 394–397. IEEE, 2013.

[9] D. Grois, D. Marpe, T. Nguyen, and O. Hadar. Comparative assessment of h. 265/mpeg-hevc, vp9, and h. 264/mpeg-avc encoders for low-delay video applications. In *Applications of Digital Image Processing XXXVII*, volume 9217, page 92170Q. International Society for Optics and Photonics, 2014.

[10] S. Han, H. Mao, and W. J. Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149*, 2015.

[11] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[12] G. Hinton, O. Vinyals, and J. Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

[13] J. Kim, J. Kwon Lee, and K. Mu Lee. Accurate image super-resolution using very deep convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1646–1654, 2016.

[14] B. Lim, S. Son, H. Kim, S. Nah, and K. M. Lee. Enhanced deep residual networks for single image super-resolution. In *The IEEE conference on computer vision and pattern recognition (CVPR) workshops*, volume 1, page 4, 2017.

[15] M. S. Sajjadi, R. Vemulapalli, and M. Brown. Frame-recurrent video super-resolution. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6626–6634, 2018.

[16] K. Winstein, A. Sivaraman, H. Balakrishnan, et al. Stochastic forecasts achieve high throughput and low delay over cellular networks. In *NSDI*, volume 1, pages 2–3, 2013.

[17] H. Yeo, S. Do, and D. Han. How will deep learning change internet video delivery? In *Proceedings of the 16th ACM Workshop on Hot Topics in Networks*, pages 57–64. ACM, 2017.

[18] H. Yeo, Y. Jung, J. Kim, J. Shin, and D. Han. Neural adaptive content-aware internet video delivery. In *OSDI*, pages 645–661, 2018.