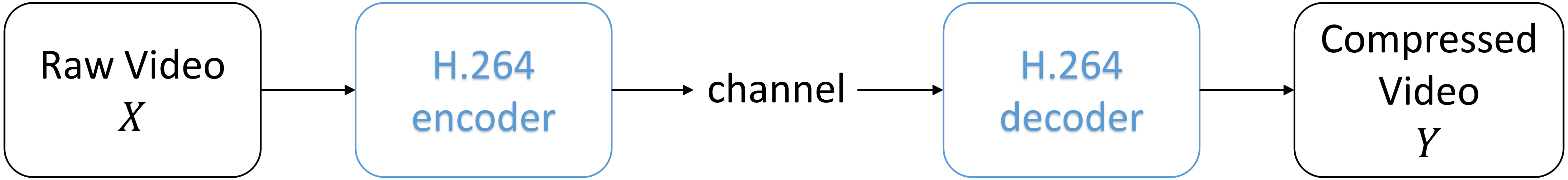
Residual Encoding for Domain-specific Video

杜俊毅 B04504028
黃漢威 B03611035

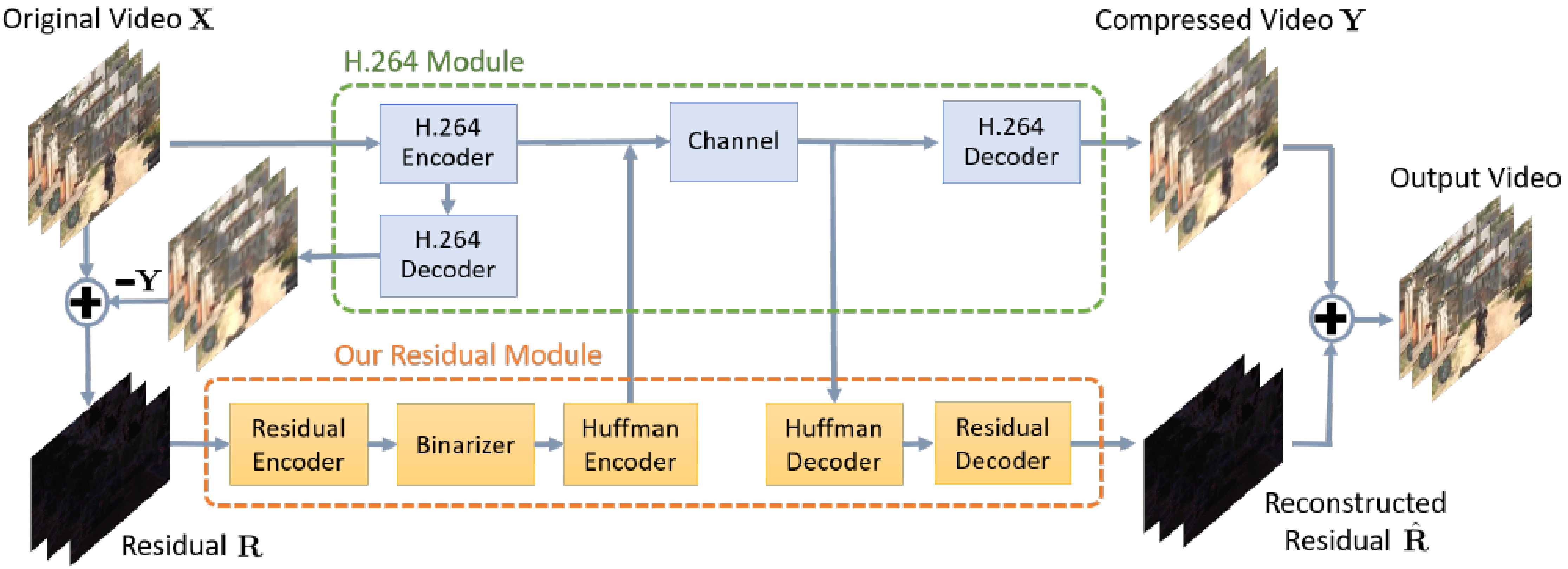
Tsai, Y.H., Liu, M.Y., Sun, D., Yang, M.H., Kautz, J.: Learning binary residual representations for domain-specific video streaming. In: AAAI

Problem Definition



- Residual = Raw – Compressed Video $R = X - Y$
- Goal: Minimize R without lowering H.264 compression ratio
  - Lowering streaming bitrate
  - Increasing video quality
  - Focusing on specific content

Algorithm


Original Video $\mathbf{X}$
H.264 Module
Compressed Video $\mathbf{Y}$
H.264 Encoder
Channel
H.264 Decoder
H.264 Decoder
Output Video
$-\mathbf{Y}$
Our Residual Module
Residual Encoder
Binarizer
Huffman Encoder
Huffman Decoder
Residual Decoder
Residual $\mathbf{R}$
Reconstructed Residual $\hat{\mathbf{R}}$

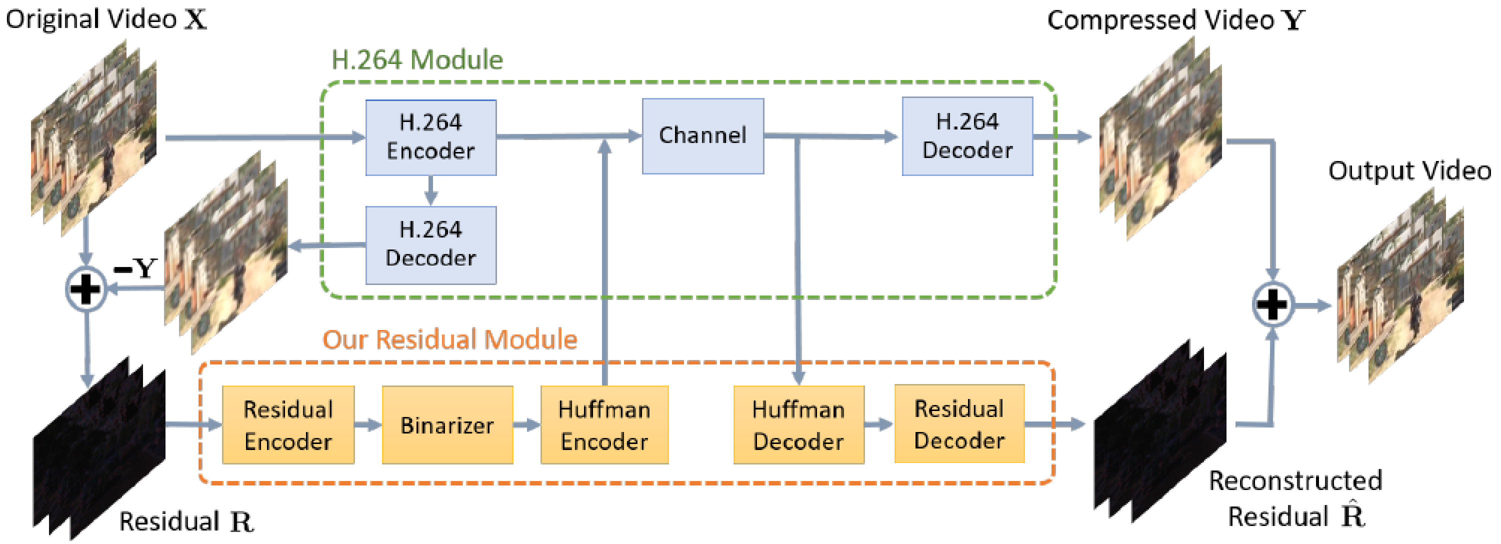
Binary Residual Autoencoder

Autoencoder consists of encoder ε, binarizer β and decoder D

Encoder: extract feature representation for binarizer

Binarizer: convert the output from encoder into a binary map

Decoder: up-sample the binary map back to the original input



Binary Residual Autoencoder

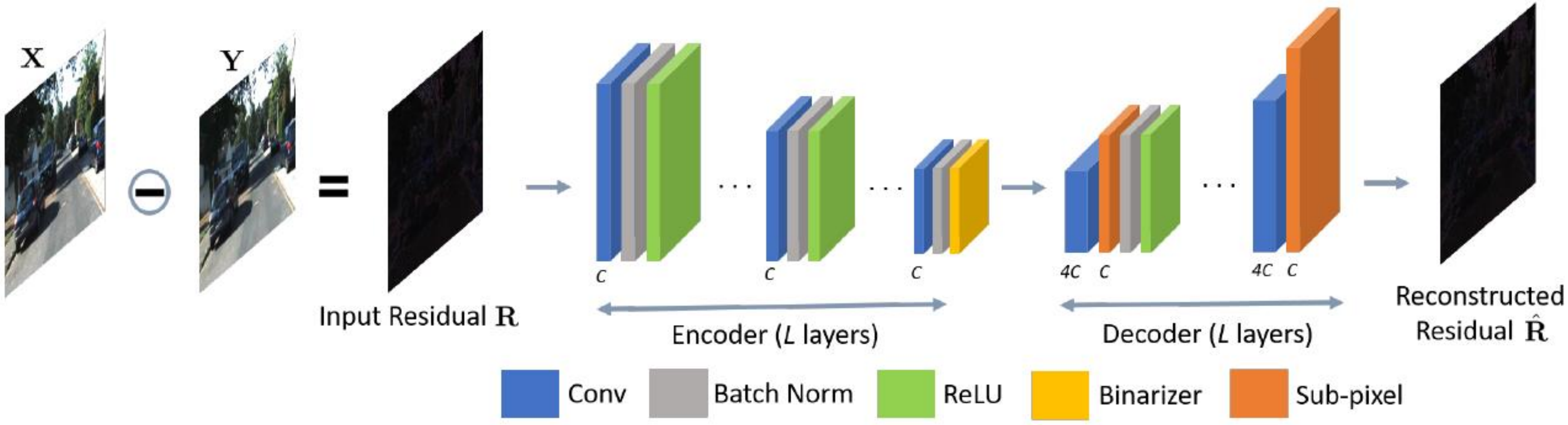
Encode and decode residual signal $\mathbf{R}$ frame by frame.

Let $\{r_i\}$ be a set of residual frames after applying H.264.

Objective function: $min_{D,\varepsilon} \sum ||r_i - D(\beta(\varepsilon(r_i)))||^2$

Sub-pixel layer: used for up-sampling [1] Shi et al 2016

Batch normalization and ReLU: facilitate the learning process.



[1] https://arxiv.org/abs/1609.05158

Binarizer

Let output feature of encoder be $e_i = \varepsilon(r_i)$

Applying activation: $z = \sigma(e_i)$, where σ can be tanh or hardtanh.

$$b(z) = \begin{cases} 1, & \text{if } z \geq 0 \\ -1, & \text{if } z < 0, \end{cases}$$

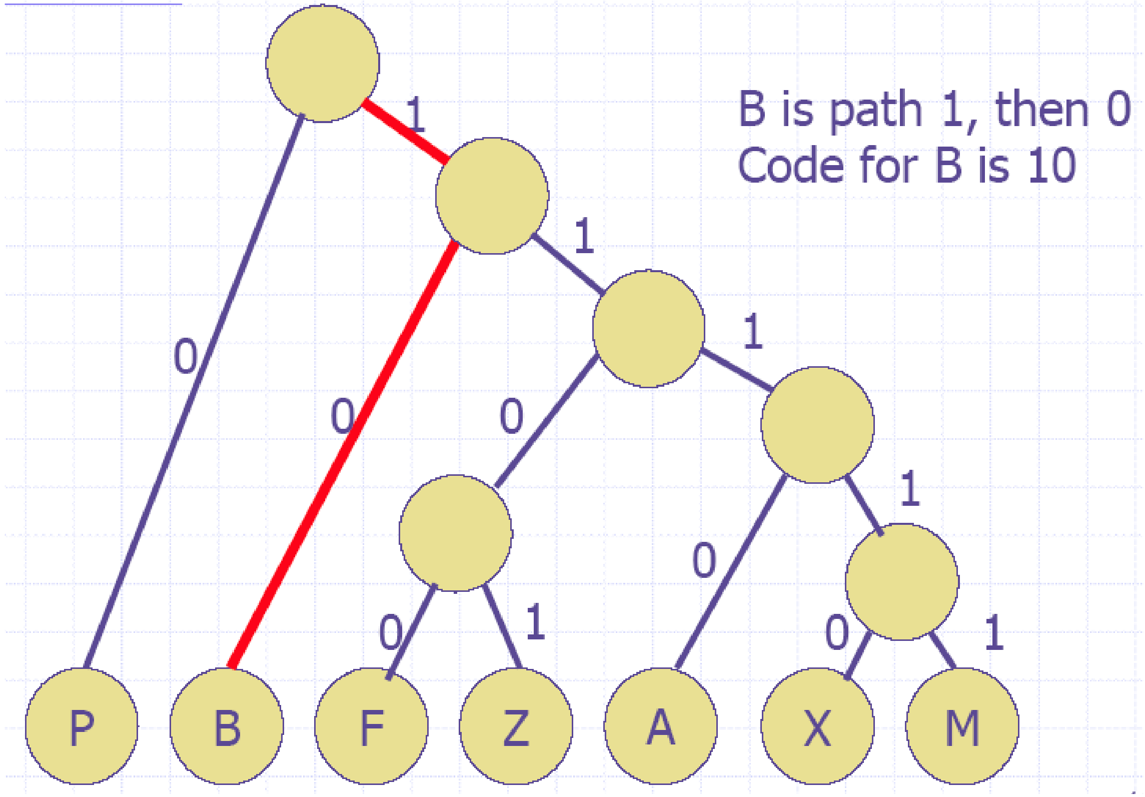
However, since binarization is not differentiable,
we cannot train the autoencoder by back-propagation.

Adopting piecewise function b_{bp} during back-propagation.

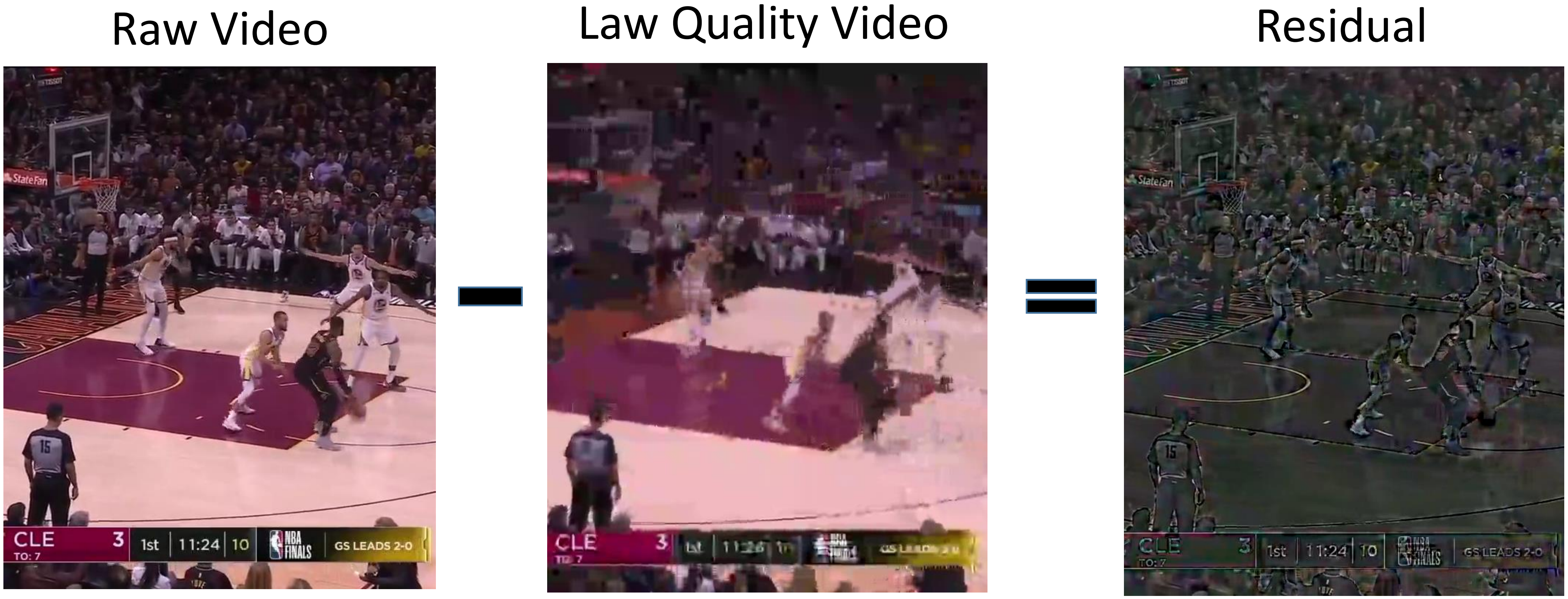
$$b_{bp}(z) = \begin{cases} 1, & \text{if } z > 1 \\ z, & \text{if } -1 \leq z \leq 1 \\ -1, & \text{if } z < -1. \end{cases}$$

Lossless Compression

After generating the binary feature map, we use lossless compression to reduce the size of the binary representation: Huffman coding



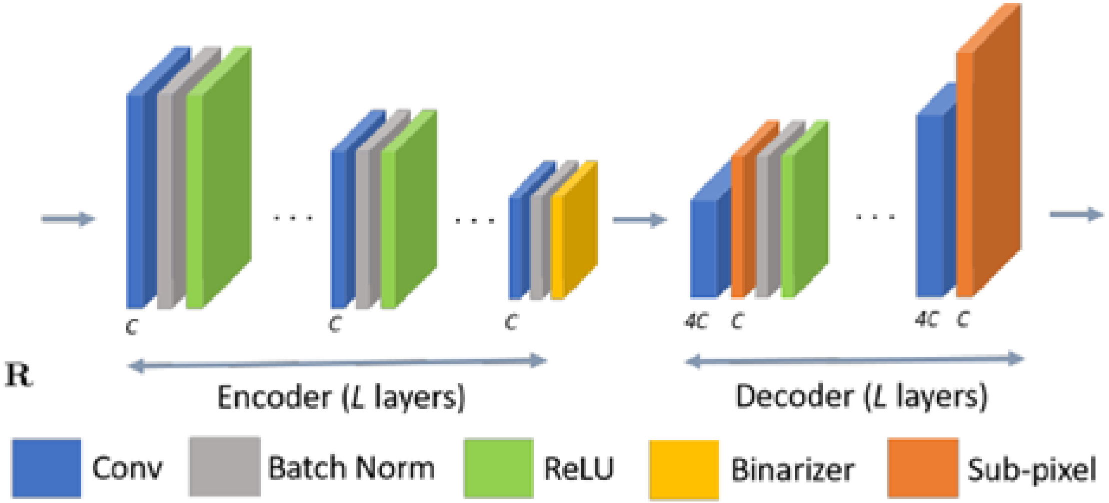
Training Data: Residual



Raw Video − Law Quality Video = Residual

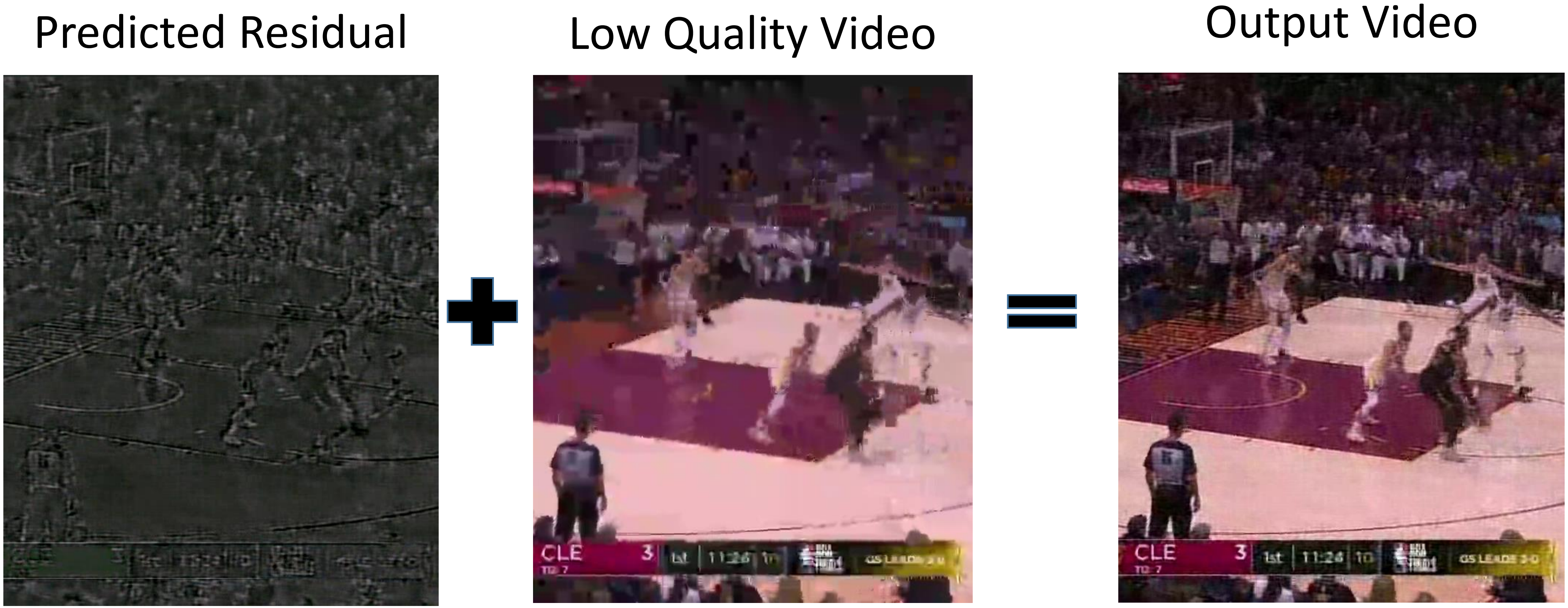
Train Autoencoder by Minizing L2-norm

True Residual



Predicted Residual

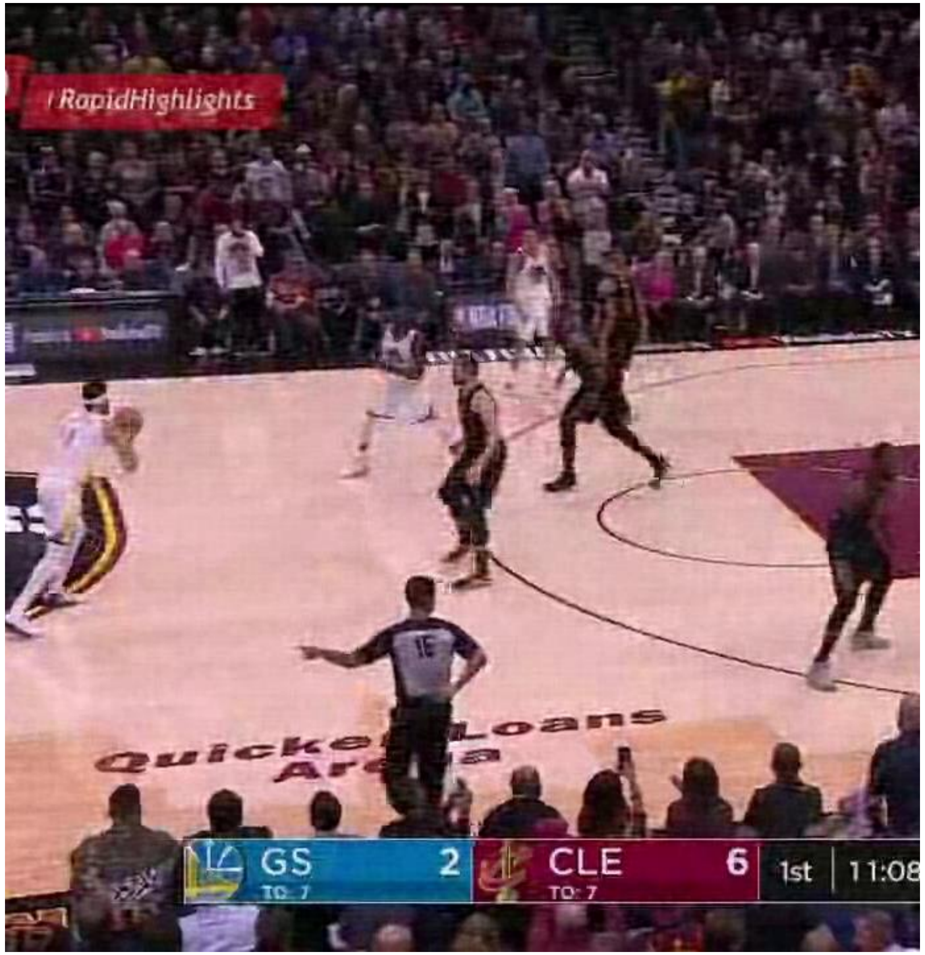
Reconstruct Output Video with Trained Model



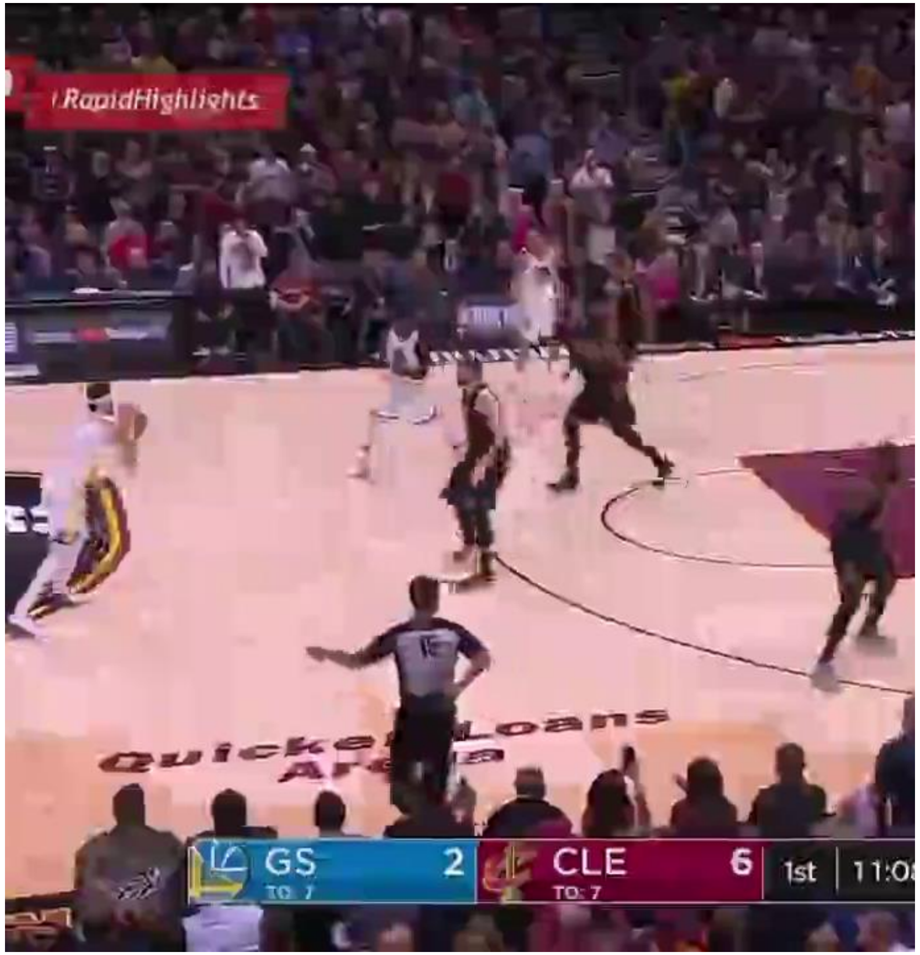
Predicted Residual + Low Quality Video = Output Video

Compare PSNR Scores

Output Video: PSNR 25



Mid-quality Video: PSNR 22

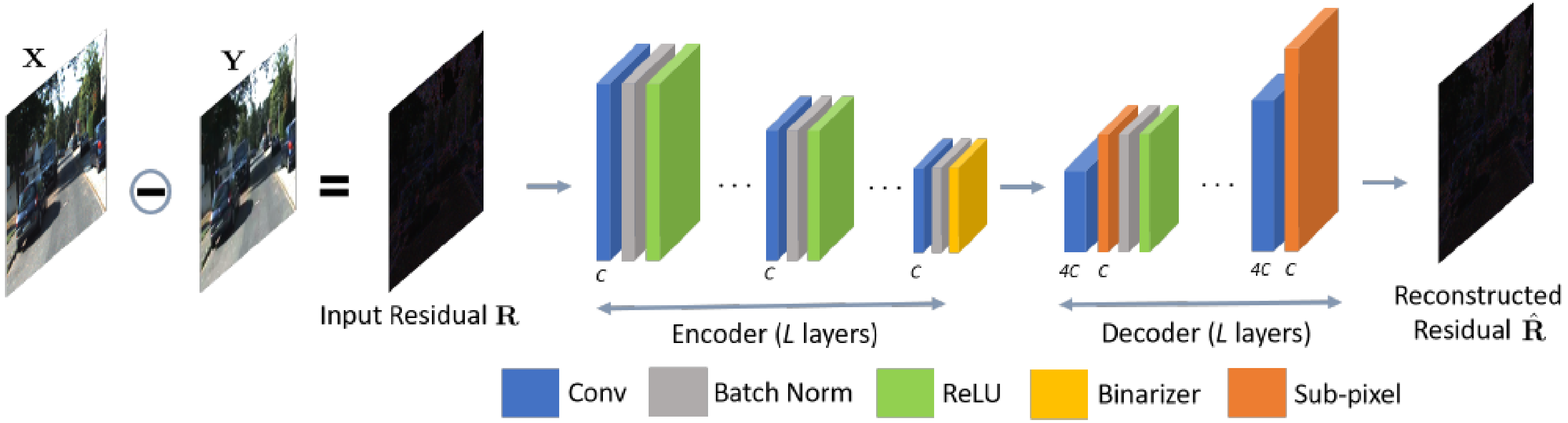


Experiment Settings

- Training set: 50 clips of volleyball spike
  - https://www.youtube.com/watch?v=NxyibrdBZ1Q
- Testing set: one clip of volleyball spike (not in the training set)
  - https://www.youtube.com/watch?v=mtpx0PjrjoY
- Three bitrates:
  - 2300kbps as ground truth
  - 1600kbps as mid quality video
  - 1200kpbs as low quality video
- Apply our residual network to the low quality video; compare the PSNR and vision quality with the ground truth and mid quality video.

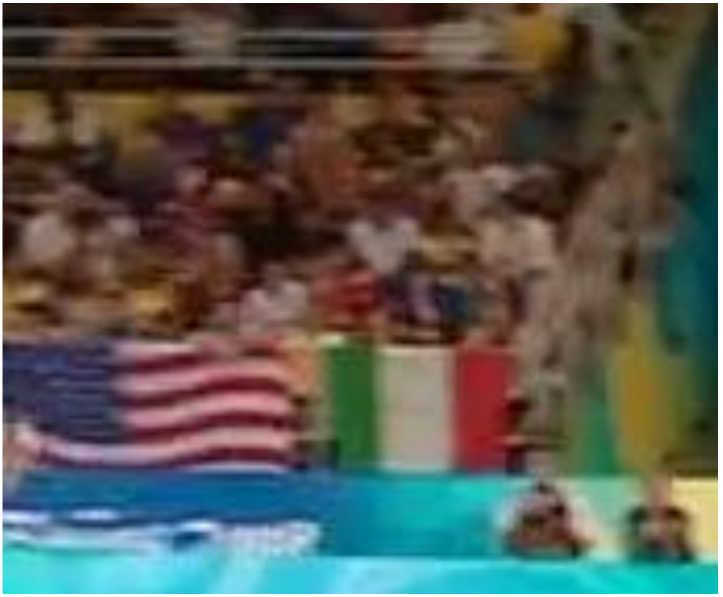
Experiment Settings

- # of CNN filters = 16
- # of CNN layers = 3
- Batch_size = 10 frames
- Adam Optimizer
- 41 epochs

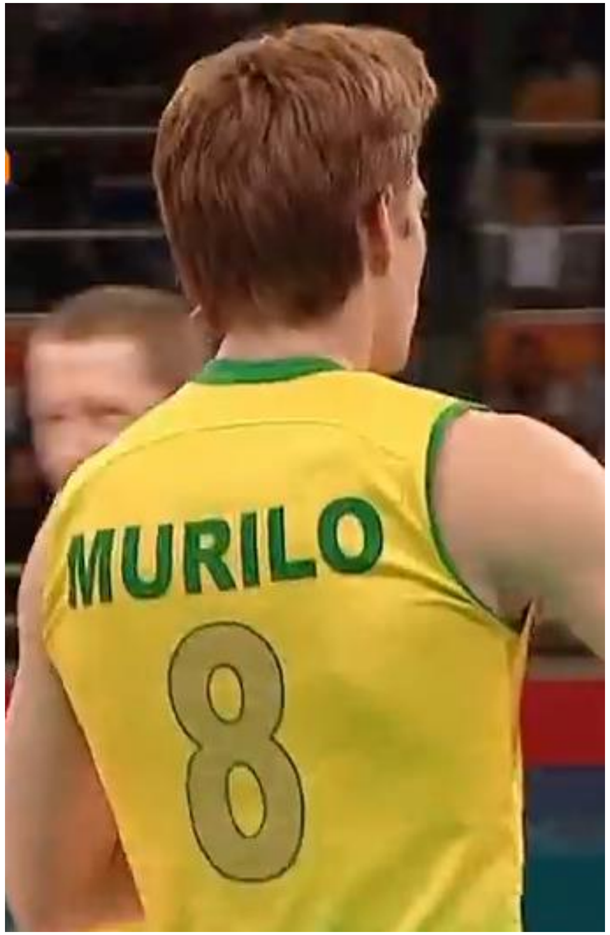
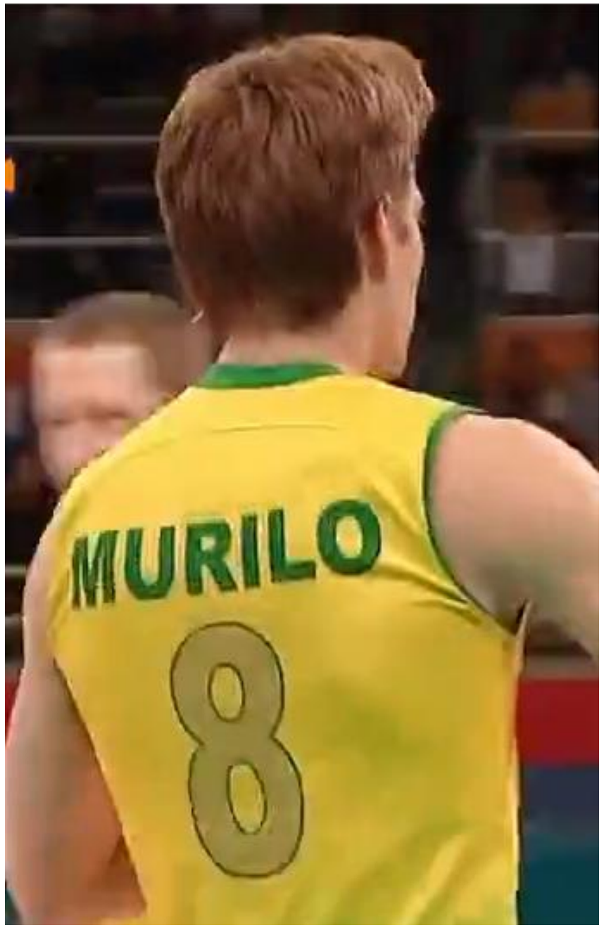
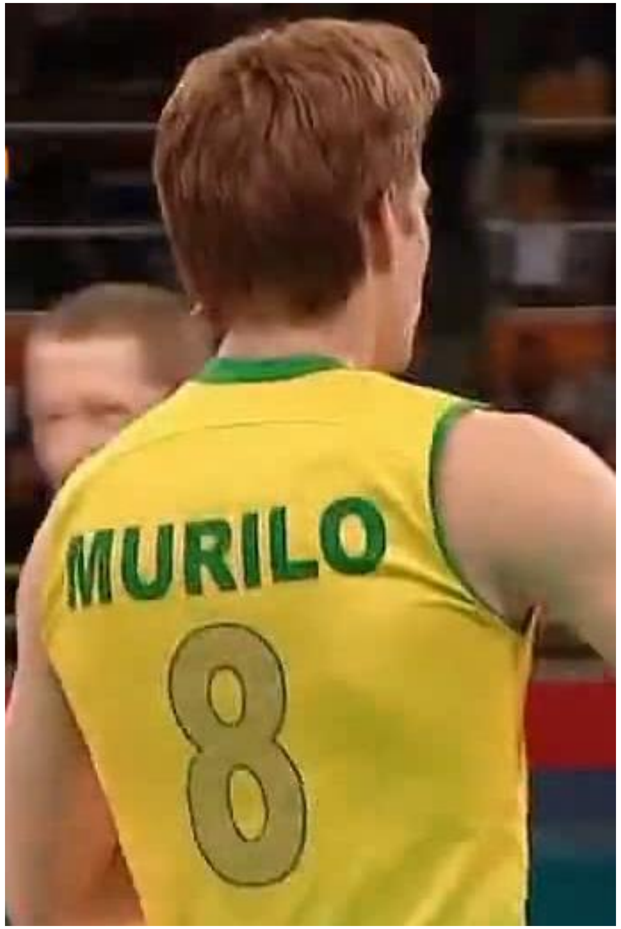


Results

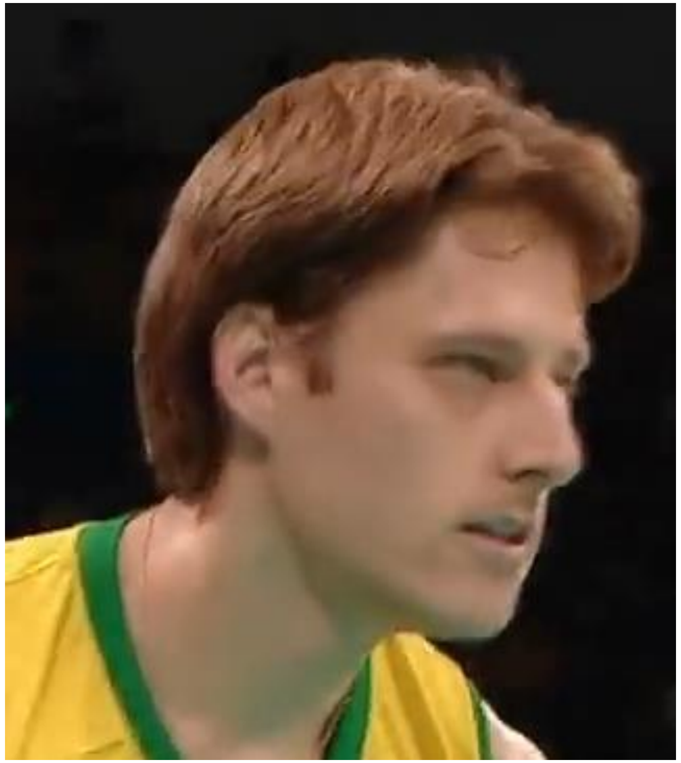
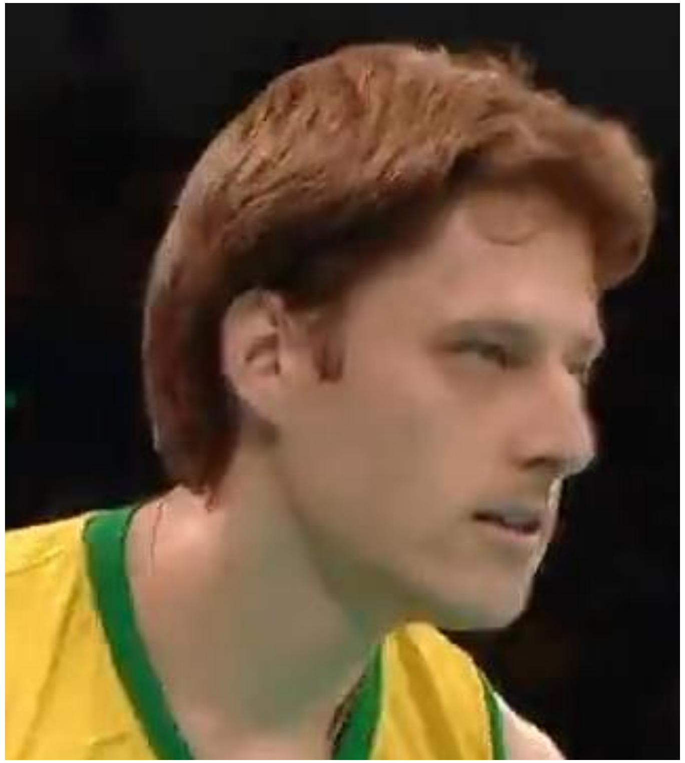
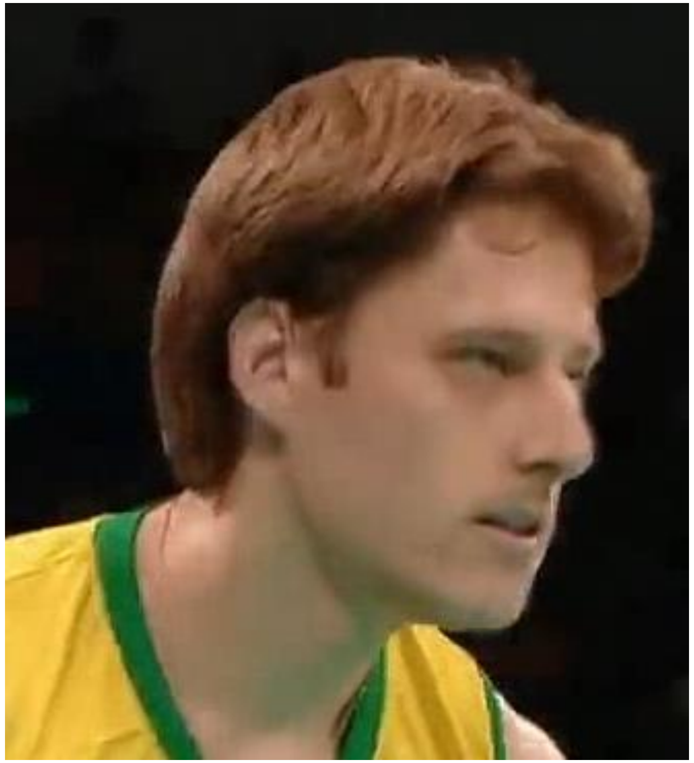
Ground Truth(2300kbps)	Mid(1600kbps)	Our result(1200kbps + residual network)



Results

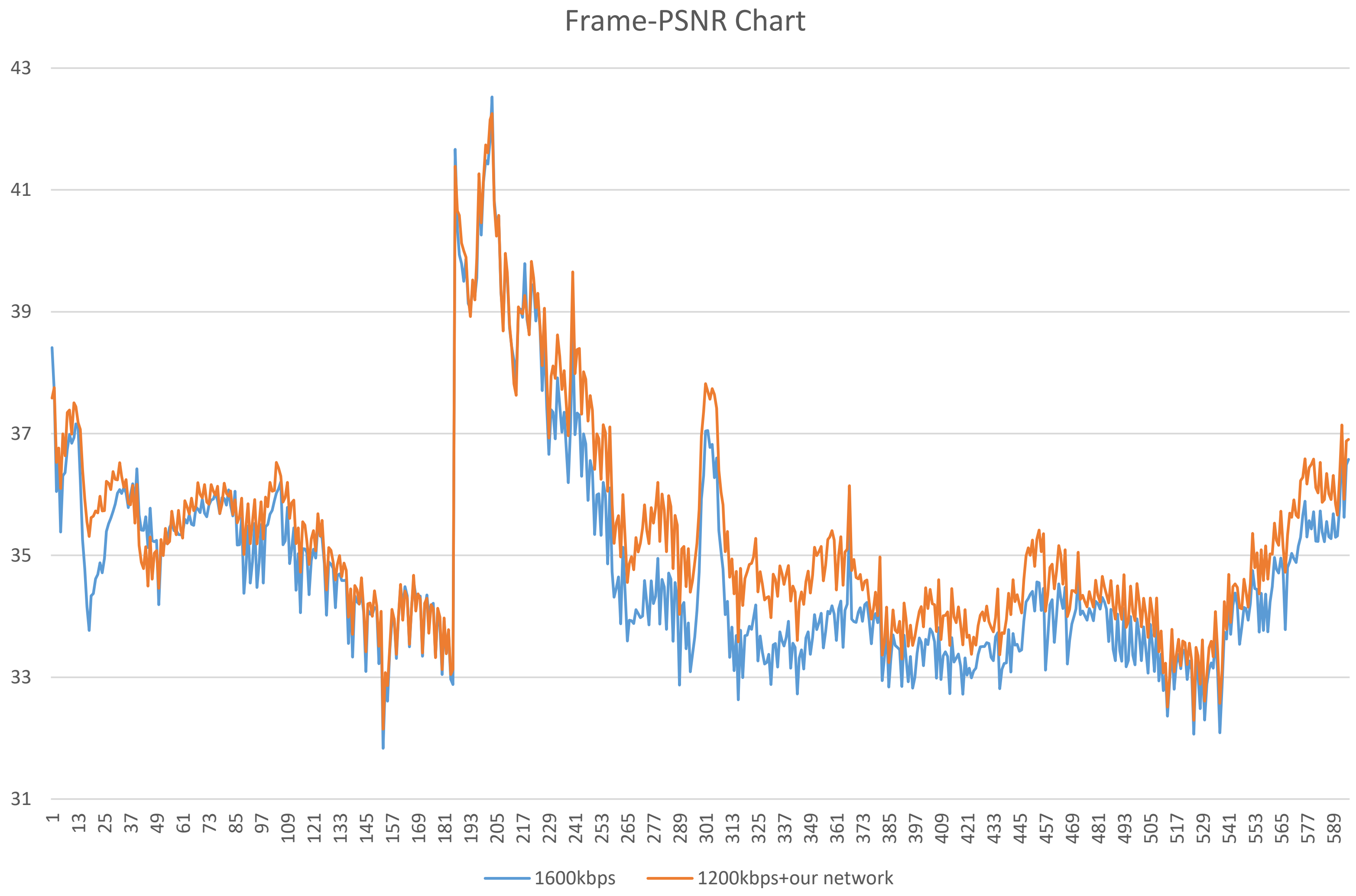
Ground Truth(2300kbps)	Mid(1600kbps)	Our result(1200kbps + residual network)
		

Results

Ground Truth(2300kbps)	Mid(1600kbps)	Our result(1200kbps + residual network)
		

Results

- Vision quality is fine.
- Encoder/Decoder size is small(<150KB).



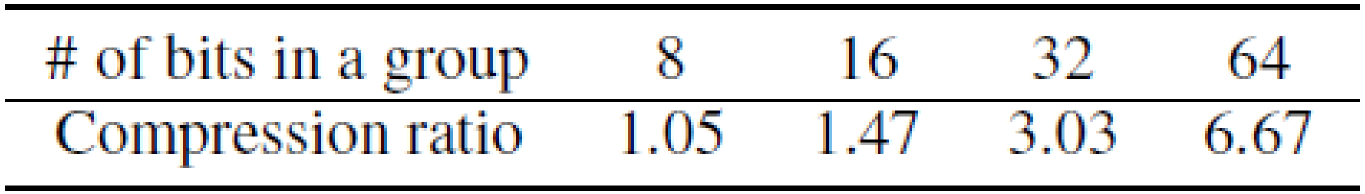
Results

- Bitmap size per sec

$$= \frac{(W * H * C)}{2^{2*L}} * fps$$

$$= \frac{1280 * 720 * 16}{2^{2*3}} * 30 = 6.59\ Mbps$$

- Applying Huffman coder with 8 bits in a group, the bitrate gets down to 3.6Mbps
- According to the paper, if we set 64 bits in a group, we can get less than 1 Mbps for residual.
- However, it’s pricey.



# of bits in a group	8	16	32	64
Compression ratio	1.05	1.47	3.03	6.67

Discussions

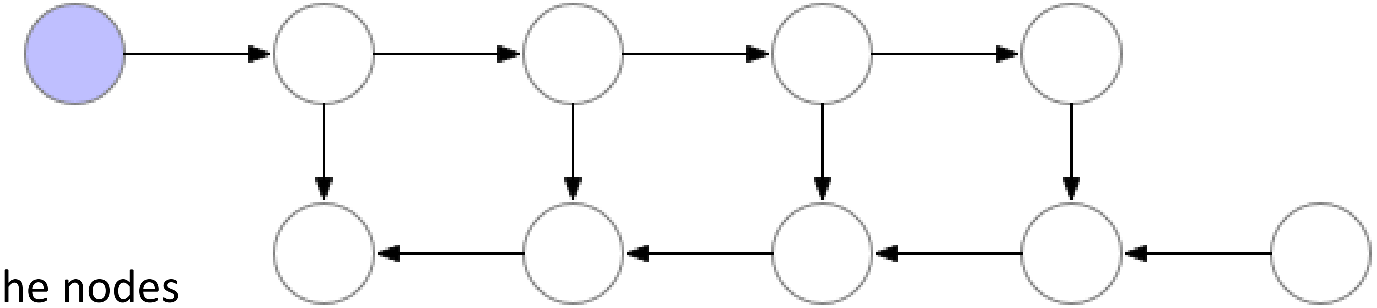
- Although the bit rate can be reduced by the autoencoder while maintaining reasonable visual quality, the author fails to consider the size of Huffman codec.
- When number of bits in a group increases, though residual size decreases, Huffman codec size sharply increases.

Bits # in a Group	8	16	32	64	128
Residual Size (Mbps)	3.17	3.09	2.15	1.04	0.496
Huffman Codec Size (MB)	0.00576	2.17	138	125	112

For 30s Video

Impractical to handle high-definition Video

- GPU memory is insufficient for high-definition Video.
- For 4K video, there are 3840×2160×3 elements in one frame. If we use 32 bits float and set batch # as 10 to train model, the size of input layer is 1GB. Similarly, the following layers output are also large.
- In backpropagation, the model usually requires saving each layers' output in memory for automatic differentiation. For deep learning, it requires more than 10GB memory.



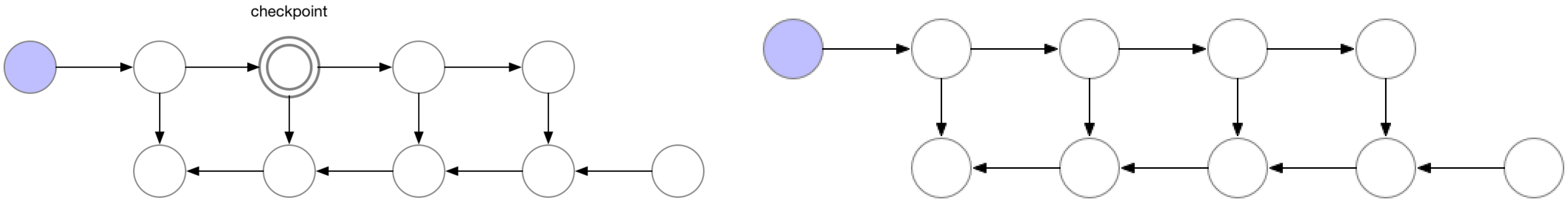
Memory required by simple backprop.
The purple shaded circles indicate which of the nodes need to be held in memory at any given time

https://github.com/openai/gradient-checkpointing

Solution?

- Using Checkpoint or reducing batch size may help, but training time will sharply increase

- However, even for 720P or 1080P video, the client without GPU is not possible to decode residual, and reconstruct video in time.



https://github.com/openai/gradient-checkpointing

Other Difficulties

- Hard to find high quality (sport) videos online
  - We can record by ourselves, but limited to local, 系隊-level games.

Conclusion

- Satisfying vision quality requires transmitting a codec first(decoder size is small). It may get more and more helpful for longer videos.
- Without powerful hardware, this method is impractical.