This supplementary material contains three appendixes. Appendix A provides the proof of Eq. 6 omitted from the main text. Appendix B shows the implementation of the 2nd order differential modulation and compares it with 1st order. Appendix C gives more experimental results on HEVC Class C and Class D data with some extra discussions.

A Proof of the Gradients Computing (Eq. 6)

We use same notations as main text. According to Algorithm 1, the updating equalities of 1st order differential modulation are divided in two steps (Eq. A.1 and A.2),

$$y_t[i] = y_t[i-1] + c_t[i] - b_t[i-1] \quad \text{(1st order differential modulation)} \tag{A.1}$$
$$b_t[i] = Q_b(y_t[i]) = B(\tanh(y_t[i])) \quad \text{(trainable quantizer)} \tag{A.2}$$

where $B(x)$ is the defined binary function which has derivative 1. So the gradients of $b_t[i]$ according to $y_t[i]$ can be computed as,

$$\frac{\partial b_t[i]}{\partial y_t[i]} = 1 - \tanh^2(y_t[i]) \tag{A.3}$$

According to A.1, $c_t[i]$ is independent with $y_t[i-1]$, so we can firstly compute the gradients of $y_t[i]$ to $y_t[i-1]$ as

$$\begin{aligned} \frac{\partial y_t[i]}{\partial y_t[i-1]} &= 1 - \frac{\partial b_t[i-1]}{\partial y_t[i-1]} \\ &= \tanh^2(y_t[i-1]) \quad \text{(by A.3)} \end{aligned} \tag{A.4}$$

According to A.3 and A.4, we firstly the gradients of $b_t[m]$ to $y_t[m-k]$,

$$\begin{aligned} \frac{\partial b_t[m]}{\partial y_t[m-k]} &= \frac{\partial b_t[m]}{\partial y_t[m]} \cdot \frac{\partial y_t[m]}{\partial y_t[m-1]} \cdot \frac{\partial y_t[m-1]}{\partial y_t[m-2]} \cdot \ldots \cdot \frac{\partial y_t[m-k+1]}{\partial y_t[m-k]} \\ &= (1 - \tanh^2(y_t[m])) \times \prod_{i=1}^{k} \tanh^2(y_t[m-i]) \end{aligned} \tag{A.5}$$

Moreover, we can compute the gradients of $y_t[i]$ to $c_t[i]$ from A.1 as,

$$\frac{\partial y_t[i]}{\partial c_t[i]} = 1 \tag{A.6}$$

Finally, we can get,

$$\begin{aligned} \frac{\partial b_t[m]}{\partial c_t[m-k]} &= \frac{\partial b_t[m]}{\partial y_t[m-k]} \cdot \frac{\partial y_t[m-k]}{\partial c_t[m-k]} = \frac{\partial b_t[m]}{\partial y_t[m-k]} \\ &= (1 - \tanh^2(y_t[m])) \times \prod_{i=1}^{k} \tanh^2(y_t[m-i]) \end{aligned} \tag{A.7}$$

B 2nd Order Differential Quantizer

The 2nd order differential quantizer has the same framework with 1st order. We just need to replace line 3 of Algorithm 1 with B.1. Note that the modulated $y_t[i]$ is computed from previous two step $i-1$ and $i-2$, which is 2nd order.

$$y_t[i] = c_t[i] + 2(y_t[i-1] - b_t[i-1]) - (y_t[i-2] - b[i-2]) \quad \text{(mod.)} \tag{B.1}$$
$$b_t[i] = y_t[i] + e[i] \quad \text{(additive noise model.)} \tag{B.2}$$

From B.1 and B.2, we can compute the final quantized output $b_t[i]$ as B.3. Transforming B.3 into Z domain, we can get B.4. The noise transfer function (NTF) is $NTF_{2nd} = (1 - Z^{-1})^2$, which is a 2nd order shaping. In fact, from the computing flow in Fig. 1, the 2nd order is twice recurrent of the 1st order. Since $NTF_{1st} = 1 - Z^{-1}$, it's clear that the NTF_{2nd} is the square of NTF_{1st}. From this point of view, the 2nd order modulation should have better performance due to the better noise shaping to alleviate the effect of quantization noise.

$$b_t[i] = c_t[i] + e_t[i] - 2e_t[i-1] + e_t[i-2] \tag{B.3}$$
$$\Rightarrow b_t(\mathcal{Z}) = c_t(\mathcal{Z}) + (1 - \mathcal{Z}^{-1})^2 e_t(\mathcal{Z}) \quad \text{(Z domain.)} \tag{B.4}$$

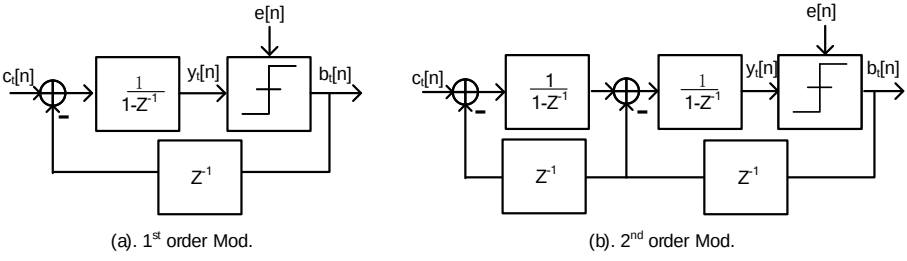


Fig. 1. Computing flow of the 1st and 2nd order differential modulations.

However, it's hard to see the improvement when 2nd order modulation is applied in deep learning. We compare the 1st and 2nd order differential modulations with experiments. The loss and evaluating results during training are given in Fig. 2. It can be observed that 2nd order modulation cannot achieve better performance than 1st. In fact, they have a similar performance in final, but 2nd order converges slower. As analyzed in the main text, the $\Delta\Sigma$ theory cannot be directly adopted into the proposed method. Our algorithm can back propagate the gradients, which is different. Regardless of the modulation order, the gradients' path are the same and the learning method should achieve a similar performance.

Fig. 2. Training loss and evaluating results for the 1st and 2nd order differential modulations.

C Performance on the HEVC Class C and Class D

The experimental results on the HEVC Class C and Class D datasets are given in Fig. 3. The proposed method performs not very well in these two datasets, especially at high BPP region. We find that our model performs not well on some videos with fast moving and motion blurs (BasketballDrill, BasketballDrive, RaceHorses , etc.). Fig. 4 gives a real example. In case of slow moving video, Fig. 4 (a) shows that although with extreme low 0.125 BPP, our method can reconstruct the detailed information from multiple frames, while H.265 has low quality. However, H.265 is better in case of fast moving video. We find fast motion between frames can cause two main problems. First, the fast movement can cause regional fuzzy image patch, and the motion blurs make the model difficult to extract features for reconstruction. Second, some scenes that are not covered in previous multiple frames may come out. This region is more difficult to reconstruct than others in our method (Fig. 4 (b) is a good example). In our current model, the optical flow and motion vectors are not adopted in the prediction network. We believe that our model can be further improved by state-of-art motion estimation methods. Nevertheless, our model still achieves high compressing quality in most videos with the novel differential modulation.

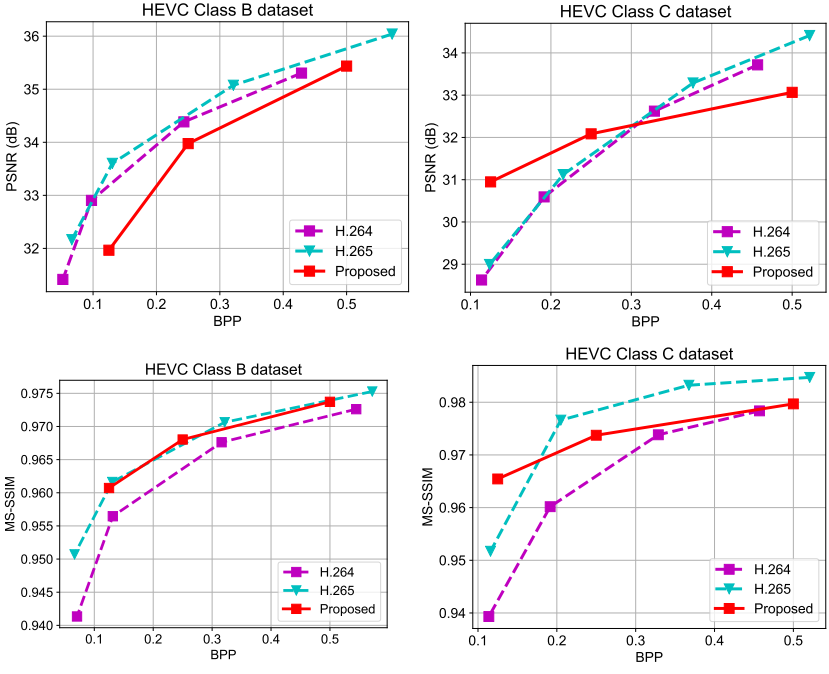
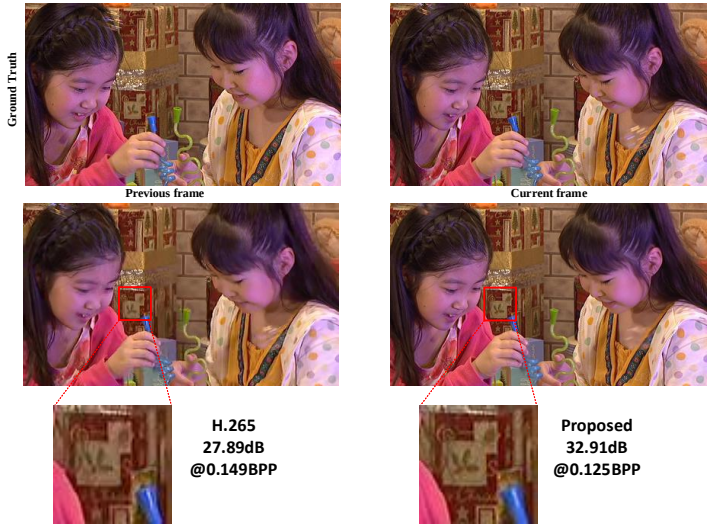


Fig. 3. Comparison of our method with H.254 and H.265 on the HEVC Class B and Class C datasets.



(a) Frames with slow moving objects.

(b) Frames with fast moving objects

Fig. 4. Performance comparison in different kinds of videos. Our current model has better performance on the videos without fast moving objects.