A Video Compression Framework Using an Overfitted Restoration Neural Network

Gang He¹,³, Chang Wu¹, Lei Li¹, Jinjia Zhou², Xianglin Wang³, Yunfei Zheng³, Bing Yu³, Weiyiing Xie¹

¹ Xi’dian University, Xi’an, China
² Graduate School of Science and Engineering, Hosei University, Tokyo, Japan
³ Beijing Kuaishou Technology

Abstract

Many existing deep learning based video compression approaches apply deep neural networks (DNNs) to enhance the decoded video by learning the mapping between decoded video and raw video (ground truth). The big challenge is to train one well-fitted model (one mapping) for various video sequences. Different with the other applications such as image enhancement whose ground truth can only be obtained in the training process, the video encoder can always get the ground truth which is the raw video. It means we can train the model together with video compression and use one model for each sequence or even for each frame. The main idea of our approach is building a video compression framework (VCOR) using overfitted restoration neural network (ORNN). A lightweight ORNN is trained for a group of consecutive frames, so that it is overfitted to this group and achieves a strong restoration ability. After that, parameters of ORNN are transmitted to the decoder as a part of the encoded bitstream. At the decoder side, ORNN can perform the same strong restoration operation to the reconstructed frames. We participate in the CLIC2020 challenge on P-frame track as the team "West-World".

1. Introduction

Recently, the deep neural networks (DNNs) based video compression approaches have developed and aroused widespread interest [5–7, 12, 18–20]. For example, the embedded hand-crafted modules are replaced with DNNs in the works [5, 7, 12, 18], such as intra prediction, motion estimation, and in-loop filter. An end-to-end deep video compression framework has been developed in [6]. The existing works considered a video compression problem being similar to other image processing problems, e.g., super resolution, denoising and deblurring, where in the real-world use case (in the testing process), the right answer (ground truth) cannot be obtained. During the training process, huge data is fed to DNN to make the output close to the ground truth (original video or the desired prediction) and make sure not overfitted to the training dataset. It is a big challenge to design a well-structured and well-trained DNN.

In this paper, we consider the unique characteristic of a video compression, where in the real-world use case, ground truth is the input raw video and can be obtained in an encoder. We propose a video compression framework named VCOR which combining a conventional video codec with an overfitted restoration neural network (ORNN). The ORNN is used to restore the conventionally compressed video in an overfitted way. At the encoder side, ORNN is trained online with the reconstructed frames from the conventional encoder as the input and the raw frames as the ground truth. Overfitting for a group of consecutive frames is activated on purpose to achieve a strong restoration ability. To transfer this ability from the encoder to the decoder, the transmitted information contains not only the syntax of a conventional codec (also called bitstream), but also the overfitted parameters of ORNN. At the decoder side, with the received parameters, an inference operation of ORNN is performed on the decoded frames to implement the strong restoration. Moreover, to achieve the performance gain by utilizing the overfitting, a lightweight ORNN is designed delicately to restore the video better, while using fewer additional transmitted parameters.

2. The proposed video compression framework

2.1. Overview of VCOR

As shown in Fig. 1, the proposed video compression framework VCOR contains the conventional compression flow and additional restoration flow, implemented by a con-
Figure 1. The proposed video compression framework VCOR contains the conventional compression and additional restoration. In an encoder, ORNN is trained online with the reconstructed frames from a conventional encoder as the input and the raw frames as the ground truth. The overfitting is realized by training it only with a group of consecutive frames. The overfitted parameters of ORNN are transmitted as a part of an encoded bitstream. In a decoder, with the transmitted parameters, an inference operation of ORNN is performed on the decoded frames to generate the final result.

2.2. Overfitted Restoration Neural Network

In VCOR, the performance gain depends on not only the restored quality, but also the additionally transmitted information. The number of parameters that affects both of them becomes critical for the design of the neural network. "The deeper, the better" is not suitable for our case. It is because although the deeper neural network can restore the video better, while times more parameters have to be transmitted. Therefore, in the proposed framework, a delicate lightweight DNN needs to be designed, which tries to achieve a better tradeoff between the quality restoration and number of parameters. Another advantage resulting from a lightweight DNN is the faster processing speed compared with other post-processing methods.

Network architecture. The overall architecture of the proposed network, which called ORNN, is illustrated in Fig. 2. Which take reconstructed target frame $\hat{I}_t$ and reference frame as input which is denoted as $HR$ and original target frame $I_t$ as label. ORNN is aimed to generate a high-quality frame $O_t$, which is close to the ground truth frame $I_t$. According to our design ideas, it can be divided into three parts, including a base branch, two multi-scale branches, and a channel attention reconstruction module. Three parts are described as follows.

(1) Base branch. We first design a very simple structure called base branch that illustrated at the bottom of Fig. 2. Stacked reconstructed frame and it’s reference frame $HR$ are directly send into the base branch, then different levels of hidden feature maps are output to reconstructed module. The whole process is defined as:

$$h_0^0, \ldots, h_n^0 = f_B(HR)$$

which $h_0^0, \ldots, h_n^0$ representing different levels of hidden feature maps. $f_B$ is our base branch network. It need to mentioned that in most of multi-frame restoration works $[1, 3, 10, 17, 20]$, neighbor frames always be explicitly aligned by motion estimation module before sent to feature extraction module. It dose improve the visual quality, while a large number of parameters must be used for this operation. It’s not economic for our system.
(2) Multi-scale branch. As mentioned above, we send reconstructed frames $HR$ directly to base branch network without any alignment operation. At the same time, due to the limitation of parameters, we are not allowed to design a very deep network. Thus our base branch has only extremely limited receptive field, which is hard to extract information form those neighbor frames with large shift, especially those import edge information. In recent years, there are many methods can be used to expand receptive fields without increase the deep of the network, such as deformable convnet [2, 22], non-local convnet [14]. But those methods also bring either times more parameters or huge amount of calculation. Inspired by works [9, 13], we choose multi-scale strategy for broader receptive field while only increase less than half the amount of calculation. Sharing network weights across scales to significantly save parameters and speed up training. To preserve edge information, we choose maxpooling for downsampling as

$$\begin{align*}
A_1 &= MaxPoolingX2(HR) \\
A_2 &= MaxPoolingX4(HR)
\end{align*}$$

where maxpooling is used for 2 times and 4 times downsampling. $A_1$ and $A_2$ are respectively send to two multi-scale branch as

$$\begin{align*}
h_0^1, \ldots, h_n^1 &= f_{MS1}(A_1) \\
h_0^2, \ldots, h_n^2 &= f_{MS2}(A_2)
\end{align*}$$

where $f_{MS1}$ and $f_{MS2}$ is our multi-scale branch network that share the same weights with the base branch. $h_0^1, \ldots, h_n^1$ and $h_0^2, \ldots, h_n^2$ are outputs of multi-scale branch, which will concatenate with $h_0^0, \ldots, h_n^0$ and then send to the channel attention reconstruction module.

(3) Channel attention reconstruction module. In our work, lots of hidden features that come from three branches are sent to reconstruction module, but generally these hidden features are not of equal importance thus attention mechanism is introduced as a guidance to bias the allocation of available features towards the most informative components. Hidden features from different branch have different scale, which need to upsample before attention

$$\begin{align*}
a_0^0, \ldots, a_0^n &= h_0^0, \ldots, h_n^0 \\
a_1^0, \ldots, a_1^n &= f_{up2}(h_0^0), \ldots, f_{up2}(h_1^n) \\
a_2^0, \ldots, a_2^n &= f_{up4}(h_0^0), \ldots, f_{up4}(h_2^n)
\end{align*}$$

where $f_{up2}$ and $f_{up4}$ are separately $\times 2$ and $\times 4$ upsampling function with pixel shuffle [11]. After upsample, all features with the same scale are concatenated then perform channel attention

$$\begin{align*}
H_{att} &= [a_0^0, \ldots, a_0^n, a_1^0, \ldots, a_1^n, a_2^0, \ldots, a_2^n] \\
F_{Att} &= f_{Att}(HR) \otimes H_{att}
\end{align*}$$

where $[,]$ and $\otimes$ separately denote concatenation and element-wise product. $f_{Att}$ is channel attention function using neural network mainly referenced methods [16, 21]. Feature $F_{Att}$ is used for final reconstruction.

$$O_t = f_{Rec}(F_{Att})$$

$f_{Rec}$ is our reconstruction function with a simple one layer convolutional neural network.

The hyper-parameter $n$ and $k$ directly influence the depth and width of our model which is influenced by specific video and can be adjust adaptively. Here we simply set it to 6 and 2.

3. Experiments

Training Details. During the training, the frames are cropped into 256x256 non-overlapping patches. The batch
The Adam optimizer\cite{4} is used by setting $\beta_1 = 0.9$ and $\beta_2 = 0.999$. The proposed ORNN is implemented with Pytorch\cite{8}. Our model contains 224.4k parameters totally. To ensure a precise backpropagation, parameters adopt 32-bit floating point format after the training to reduce the transmitted information. Table 1 shows the different model size between 32-bit floating point format and 16-bit floating point format. In CLIC competition, we use one model fit all test data.

**Loss Function.** We choose MS-SSIM\cite{15} as our loss function. Which can be described as follow.

$$l(x, y) = \frac{2\mu_x\mu_y + C_1}{\mu_x^2 + \mu_y^2 + C_1}$$

$$c(x, y) = \frac{2\sigma_x\sigma_y + C_2}{\sigma_x^2 + \sigma_y^2 + C_2}$$

$$s(x, y) = \frac{\mu_x\mu_y + C_3}{\mu_x^2 + \mu_y^2 + C_3}$$

$$\text{MS-SSIM}(x, y) = \left[ l_M \right]^\alpha \cdot \prod_{j=1}^M \left[ c_j \right]^{\beta_j} \left[ s_j \right]^{\gamma_j}$$

**References**


[21] Yulun Zhang, Kunpeng Li, Kai Li, Lichen Wang, Bineng Zhong, and Yun Fu. Image super-resolution using very deep