Task-Aware Encoder Control for Deep Video Compression

Supplementary Material

6. Implementation details of using DFS algorithm to find the optimal GoP structure for Bpp-mAP metric

Different videos have different contents, and some of the contents are related to downstream machine vision tasks, including the size and quantity of objects, intensity of object movement, camera movement, etc. Considering these factors related to coding and downstream tasks, when trading off between the coding bitrate and downstream task performance, different videos must correspond to different optimal GoP structures. To explore the "optimal GoP structures" and the upper bound of GoP structures using P and P_m frames, as shown in algorithm 1, we use a DFS algorithm to iterate over every possible structure for every GoP, which has a size of 10. In every GoP, we use the target function $R + \lambda L_{det}$ and find the GoP structure that leads to the minimum value of the target function. As shown in Fig. 5, the result of DFS shows a much better trade-off for the Bpp-mAP metric than the hand-craft "DivGoP" structure. However, the time complexity of DFS is extremely high. For GoP size 10, there are 9 predicted frames, which means that the algorithm will visit $2^9 = 512$ nodes, and 512 times of encoding and decoding will be performed in each GoP. This obviously cannot be really applied. At the same time, the result of DFS does show that there is a huge gap between the hand-craft "DivGoP" structure and optimal GoP structure. Therefore, how to use the inter-frame relationship, motion relationship and other information of the videos to dynamically determine the GoP structures more accurately within acceptable time complexity is a worth exploring question for us.

Algorithm 1 Using DFS to search for the optimal GoP structure on MOT

function FINDTARGETPATH($depth, path, min\_path, bpps, det\_losses, P_m$)
  if $depth > max\_depth$ **then**
    return
  end if
  if $P_m == True$ **then**
    $bpp, det\_loss \leftarrow codec.forward\_P_m$
    $path.append(0)$
    $bpps.append(bpp)$
    $det\_losses.append(det\_loss)$
  else
    $bpp, det\_loss \leftarrow codec.forward\_P$
    $path.append(0)$
    $bpps.append(bpp)$
    $det\_losses.append(det\_loss)$
    ▷ Update Reference frame
  end if
  if $depth == max\_depth$ **then**
    $target \leftarrow target\_function(bpps, det\_losses)$
    if $target < min\_target$ **then**
      $min\_path \leftarrow current\_path$
      $min\_target \leftarrow target$
    end if
  end if
  FINDTARGETPATH($depth + 1, ..., P_m = True$)
  FINDTARGETPATH($depth + 1, ..., P_m = False$)
end function

7. Comparing with other one-to-one VCM methods

In this section, we compare our method with existing one-to-one VCM methods in multi-object tracking and video object detection tasks.

For multi-object tracking task, we compare our method with SMC [38], which is a "one-to-one" VCM method. SMC contains two layers of encoders and decoders. The base layer is VVC and the semantic layer is designed to enhance the performance for downstream vision tasks. We requested the original authors of SMC to obtain the code and rigorously compared with it in our experimental settings. The results are shown in Fig. 10(a)(b)(c)(d). We compare our method with SMC in the metrics of MOTA, MOTP, IDF1 and FN. It is observed that out methods outperform SMC at bitrate range from around 0.04 to around 0.14. Our controlled TCM and controlled FVC still show the best and second-best performance.

We also compare our method with DeepSVC [23]. DeepSVC contains three layers of encoders and decoders, which are semantic layer, structure layer and texture layer. The semantic layer serves downstream vision tasks, while structure and texture are used for the reconstruction task.

As for the video object detection task, we follow the experimental setting of DeepSVC, using the same detector TROIAlign [13] whose pre-trained weights are provided by mmtracking [6]. Specifically, we follow the data processing procedure of DeepSVC and compress the video frames using our method, then we feed the decoded frames to TROIAlign and evaluate the video object detection performance. The results are shown in Fig. 10(e). The "Ours+TCM" method outperforms DeepSVC in this Bpp-mAP metric. Compared with DeepSVC, our method has

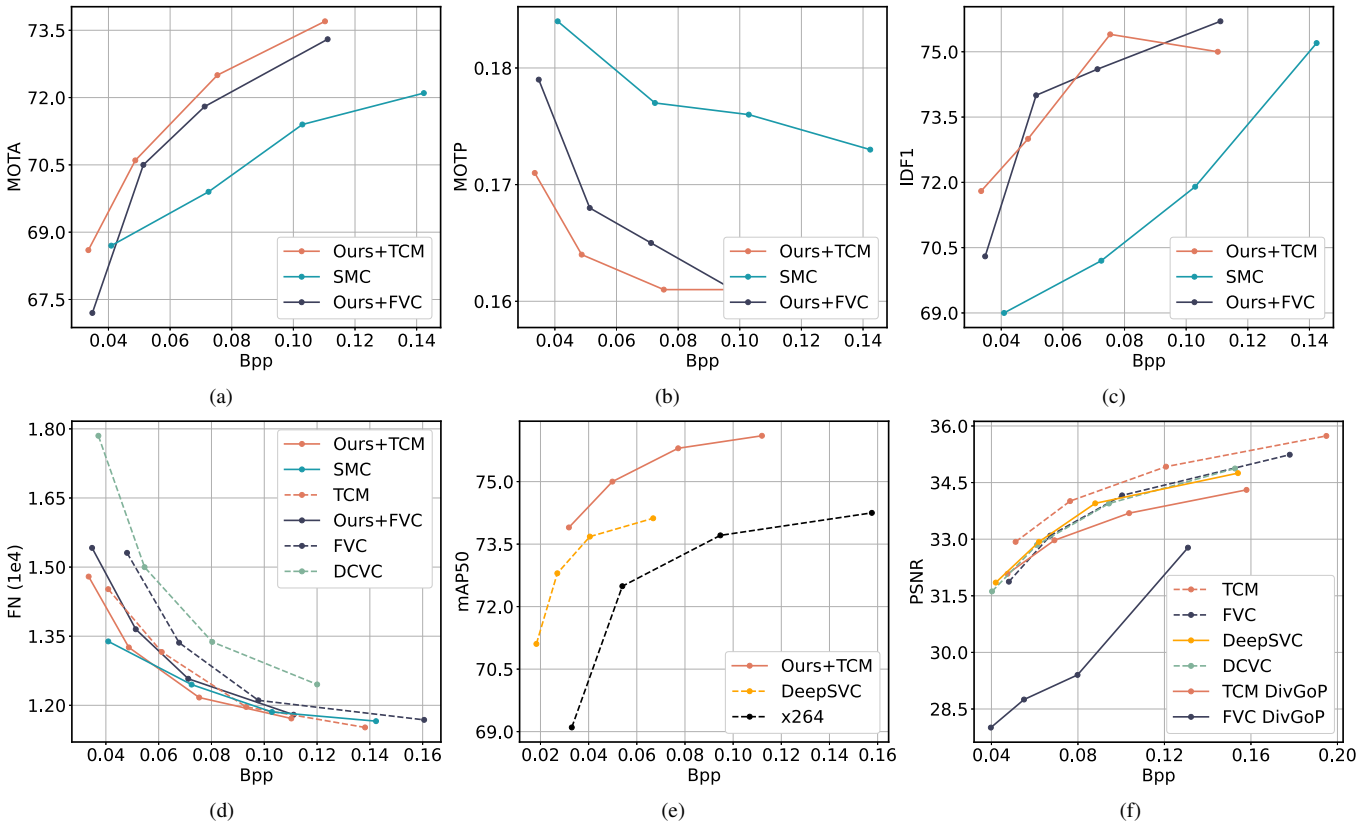

Figure 10. (a) Bpp-MOTA curves on MOT Dataset. (b) Bpp-MOTP curves on HEVC ClassB dataset. (c) Bpp-IDF1 curves on MOT Dataset. (d) Bpp-FN curves on MOT Dataset. (e) Bpp-mAP curves on Imagenet VID dataset. (f) Bpp-PSNR curves on HEVC ClassB dataset.

two main strengths, on the one hand, our method is built on existing DVC codecs, making it both flexible and have the potential to use stronger DVC codecs to further demonstrate stronger performance. On the other hand, many video analysis methods use reference frame sampling strategy to enhance the video analysis performance. Some One-to-one VCM methods, such as DeepSVC, use feature decoder to support the video analysis task, making it difficult to still support the reference frame sampling strategy. Meanwhile, Our method is compatible with this strategy since we do not change the decoding procedure and still generate decoded video frames.

As for the video reconstruction task, we compare our methods with DeepSVC on HEVC Class B dataset. The result in Fig. 10(f) shows that our TCM still outperforms DeepSVC in Bpp-PSNR metric.

Our codec shows clear advantages over recent VCM methods such as SMC and DeepSVC: (1) Both SMC and DeepSVC require multiple decoders for different tasks, whereas our method necessitates only a single pre-trained decoder to support multiple tasks, which is more feasible to standardization and practical deployment. Once standardized and deployed in SoC, the decoder remains unchanged, allowing our codec to support new tasks without modifications. (2) As shown in Fig. 10, our method surpasses these competitors in MOT and VOD tasks in performance.

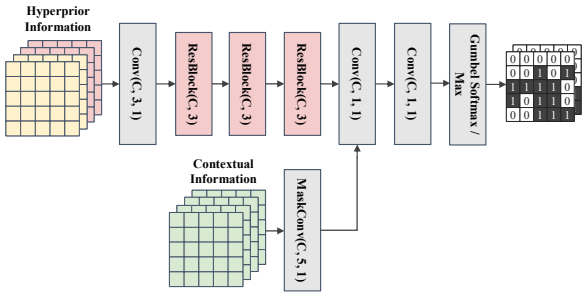

Figure 11. Autoregressive DVMP module.

8. Detail structure of autoregressive DVMP module

As illustrated in Fig. 11, we provide the structure of autoregressive DVMP module. The hyperprior network then pre-

dicts the mean and variance for each element, resulting in dimensions of $2c \times h \times w$. And the autoregressive networks predicts the contextual feature, which also has a shape of $2c \times h \times w$. The autoregressive DVMP firstly obtain the hyperprior information, undergoing one convolution layer and three ResBlocks with kernel size of 3. Then the autoregressive convolution layer is used to handle the contextual information. After that, the two parts of feautures are concatenated and undergo two convolution layers with kernel size of 1. In this way, our DVMP can achieve the support for mode prediction in DCVC. During training stage, we employ the Gumbel Softmax technique to predict the m_t. And during inference stage, we use max optain to generate the m_t.

9. Discussion of extract and compress semantic information instead of whole video.

The idea of extracting and compressing semantic information to support downstream machine vision tasks is feasible, as evidenced by several existing methods [4, 11]. Nonetheless, in complex downstream scenarios, relying solely on compressed semantic information may be insufficient. Take autonomous driving as an example: multimodal data, such as video, LIDAR, and radar, are imperative for precise decision-making. While semantic information like object detection is crucial, the video modality encompasses a broader context — including weather conditions and road quality — which can profoundly impact driving decisions. Furthermore, in practical applications, retaining video modality for purposes like secondary human inspection post-detection is often necessary. Therefore, our study maintains a focus on the video modality.