AN END-TO-END VIDEO COMPRESSION USING DEEP NEURAL NETOWRK

B. Shravan Kumar[1], Dr. V. Usha Shree[2]
[1]Research Scholar JNTUH Hyderabad, India
[2]Professors and Principal, JBREC, R.R Dist, India

ABSTRACT

Conventional video compression approaches utilize the predictive coding engineering and encode the relating motion data and remaining data. In this paper, exploiting both traditional engineering in the conventional video compression technique and the incredible non-direct portrayal capacity of neural organizations, we propose the principal start to finish video compression profound model that mutually upgrades all the parts for video compression. In particular, learning based optical stream assessment is used to acquire the motion data and reproduce the current casings. At that point we utilize two auto-encoder style neural organizations to pack the relating motion and lingering data. All the modules are mutually learned through a solitary misfortune work, in which they team up with one another by considering the compromise between diminishing the quantity of compression bits and improving nature of the decoded video. Exploratory outcomes show that the proposed approach can beat the generally utilized video coding standard H.264 regarding PSNR and be even comparable to the most recent standard H.265 as far as MS-SSIM.

INTRODUCTION

These days, video content adds to over 80% web traffic [26], and the rate is required to increment much further. Accordingly, it is basic to manufacture a productive video compression framework and generate greater casings at given transmission capacity financial plan. Also, most video related PC vision errands, for example, video object identification or video object following are touchy to the nature of compacted videos, and productive video compression may bring benefits for other PC vision undertakings. Then, the methods in video compression are additionally useful for activity acknowledgment [41] and model compression [16].

In any case, in the previous many years, video compression calculations [39, 31] depend available created modules, e.g., block-based motion assessment and Discrete Cosine Transform (DCT), to diminish the redundancies in the video successions.

Albeit every module is all around planned, the entire compression framework isn't start to finish advanced. It is alluring to additionally improve video compression execution by mutually upgrading the entire compression framework. As of late, deep neural organization (DNN) based autoencoder for picture compression [34, 11, 35, 8, 12, 19, 33, 21, 28, 9] has accomplished equivalent or far superior execution than the conventional picture codecs like JPEG [37], JPEG2000 [29] or BPG [1]. One potential clarification is that the DNN based picture compression strategies can abuse huge scope start to finish preparing and exceptionally non-straight change, which are not utilized in the customary methodologies. In any case, it is non-paltry to legitimately apply these strategies to fabricate a start to finish learning framework for video compression. To begin with, it stays an open issue to figure out how to generate and pack the motion data custom fitted for video compression. Video compression strategies vigorously depend on motion data to decrease worldly excess in video successions. A direct arrangement is to utilize the learning based optical stream to speak to motion data. Nonetheless, current learning based optical stream approaches target producing stream fields as accurate as could be expected under the circumstances. Yet, the exact optical stream is frequently not ideal for a specific video task [42]. Likewise, the information volume of optical stream increments essentially when contrasted and motion data in the conventional compression frameworks and straightforwardly applying the current compression approaches in [39, 31] to pack optical stream esteems will fundamentally expand the quantity of pieces needed for putting away motion data. Second, it is indistinct how to manufacture a DNN based video compression framework by limiting the rate-distortion based goal for both remaining and motion data. Rate-distortion optimization (RDO) targets accomplishing higher caliber of recreated outline (i.e., less distortion) when the quantity of pieces (or touch rate) for compression is given. RDO is significant for video compression execution. To abuse the intensity of start to finish preparing for learning based compression framework, the RDO strategy is needed to upgrade the entire framework.

PROPOSED METHODOLOGY

In this paper, we propose the primary start to finish deep video compression (DVC) model that together learns motion estimation, motion compression, and lingering compression. The benefits of this network can be summed up as follows:

• All critical segments in video compression, i.e., motion estimation, motion pay, leftover compression, motion compression, quantization, and touch rate estimation, are executed with a start to finish neural network.

• The critical segments in video compression are together improved dependent on rate-distortion compromise through a solitary misfortune work, which prompts higher compression productivity.

• There is coordinated planning between the parts of conventional video compression draws near and our proposed DVC model. This work fills in as the scaffold for specialists chipping away at video compression, PC vision, and deep model plan.

RELATED WORK

Image Compression

A great deal of picture compression calculations has been proposed in the previous many years [37, 29, 1]. These strategies vigorously depend on high quality methods. For instance, the JPEG standard straight maps the pixels to another portrayal by utilizing DCT, and quantizes the comparing coefficients before entropy coding [37]. One inconvenience is that these modules are separately enhanced and may not accomplish ideal compression execution.

As of late, DNN based picture compression strategies have pulled in increasingly more consideration [34, 35, 11, 12, 33, 8, 21, 28, 24, 9]. In [34, 35, 19], repetitive neural networks (RNNs) are used to fabricate a reformist picture compression plot. Different strategies utilized the CNNs to plan an auto-encoder style network for picture compression [11, 12, 33]. To streamline the neural network, the work in [34, 35, 19] simply attempted to limit the distortion (e.g., mean square blunder) between unique casings and remade outlines without considering the quantity of pieces utilized for compression.

Rate-distortion optimization method was embraced in [11, 12, 33, 21] for higher compression productivity by presenting the quantity of pieces in the optimization methodology. To assess the touch rates, setting models are found out for the versatile math coding technique in [28, 21, 24], while non-versatile number-crunching coding is utilized in [11, 33]. Likewise, different procedures, for example, summed up disruptive standardization (GDN) [11], multi-scale picture disintegration [28], ill-disposed preparing [28], significance map [21, 24] and intra expectation [25, 10] are proposed to improve the picture compression execution. These current works are significant structure blocks for our video compression network.

Video Compression

In the previous many years, a few customary video compression calculations have been proposed, for example, H.264 [39] and H.265 [31]. The majority of these calculations follow the predictive coding design. Despite the fact that they give profoundly effective compression execution, they are physically planned and can't be together improved in a start to finish way. For the video compression task, a ton of DNN based techniques have been proposed for intra expectation and leftover coding [13], mode choice [22], entropy coding [30], post-handling [23]. These techniques are utilized to improve the exhibition of one specific module of the customary video compression calculations as opposed to building a start to finish compression conspire.

In [14], Chen et al. proposed a square based learning approach for video compression. In any case, it will unavoidably generate blockness antique in the limit between blocks. Furthermore, they utilized the motion data spread from past remade outlines through conventional square based motion estimation, which will corrupt compression execution.

Tsai et al. proposed an auto-encoder network to pack the leftover from the H.264 encoder for explicit space videos [36]. This work doesn't utilize deep model for motion estimation, motion remuneration or motion compression.

The most related work is the RNN based methodology in [40], where video compression is detailed as casing interjection. Be that as it may, the motion data in their methodology is likewise generated by customary square based motion estimation, which is encoded by the current non-deep learning-based picture compression strategy [5]. As such, estimation and compression of motion are not cultivated by deep model and mutually improved with different segments. Furthermore, the video codec in [40] just targets limiting the distortion (i.e., mean square blunder) between the first edge and reproduced outline without considering rate-distortion compromise in the preparation method. In correlation, in our network, motion estimation and compression are accomplished by DNN, which is together improved with different parts by considering the rate distortion compromise of the entire compression framework.

Motion Estimation

Motion estimation is a critical segment in the video compression framework. Conventional video codecs utilize the square based motion estimation calculation [39], which well backings equipment execution.

In the PC vision assignments, optical stream is broadly used to misuse transient relationship. As of late, a ton of learning based optical stream estimation techniques [15, 27, 32, 17, 18] have been proposed. These methodologies spur us to integrate optical stream estimation into our start to finish learning structure. Contrasted and square based motion estimation strategy in the current video compression draws near, learning based optical stream strategies can give accurate motion data at pixel-level, which can be likewise streamlined in a start to finish way. In any case, substantially more pieces are needed to pack motion data if optical stream esteems are encoded by conventional video compression draws near.

SYSTEM DESIGN

Introduction

By and large, the video compression encoder generates the bitstream dependent on the info current edges. Furthermore, the decoder reproduces the video outlines dependent on the got bitstreams.

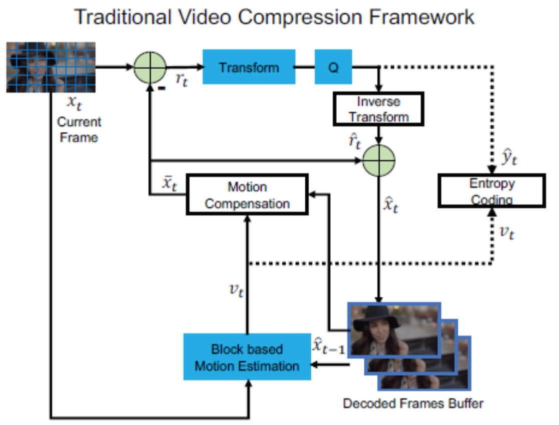


Figure 1: The predictive coding architecture used by the traditional video codec H.264 [39] or H.265 [31].

In above Fig, all the modules are remembered for the encoder side while blue shading modules are excluded from the decoder side.

The exemplary structure of video compression in Fig. 1 follows the anticipate change engineering. In particular, the info outline x_t is part into a bunch of squares, i.e., square districts, of a similar size (e.g., 8 × 8). The encoding methodology of the conventional video compression calculation in the encoder side is appeared as follows,

Step 1. **Motion estimation**. Estimate the motion between the current frame x_t and the previous reconstructed frame ˆx$_{t-1}$. The corresponding motion vector v_t for each block is obtained.

Step 2. **Motion compensation**. The predicted frame $\bar{x}_t$ is obtained by copying the corresponding pixels in the previous reconstructed frame to the current frame based on the motion vector v_t defined in Step 1. The residual rt between the original frame x_t and the predicted frame $\bar{x}_t$ is obtained as $r_t = x_t - \bar{x}_t$.

Step 3. **Transform and quantization**. The residual r_t from Step 2 is quantized to $\hat{y}_t$. A linear transform (*e.g.*, DCT) is used before quantization for better compression performance.

Step 4. **Inverse transform**. The quantized result $\hat{y}_t$ in Step 3 is used by inverse transform for obtaining the reconstructed residual $\hat{r}_t$.

Step 5. **Entropy coding**. Both the motion vector v_t in Step 1 and the quantized result $\hat{y}_t$ in Step 3 are encoded into bits by the entropy coding method and sent to the decoder.

Step 6. Frame reconstruction. The reconstructed frame $\hat{x}_t$ is obtained by adding $\bar{x}_t$ in Step 2 and $\hat{r}_t$ in Step 4, *i.e.* $\hat{x}_t = \hat{r}_t + \bar{x}_t$. The reconstructed frame will be used by the (t + 1)th frame at Step 1 for motion estimation.

For the decoder, based on the bits provided by the encoder at Step 5, motion compensation at Step 2, inverse quantization at Step 4, and then frame reconstruction at Step 6 are performed to obtain the reconstructed frame $\hat{x}_t$.

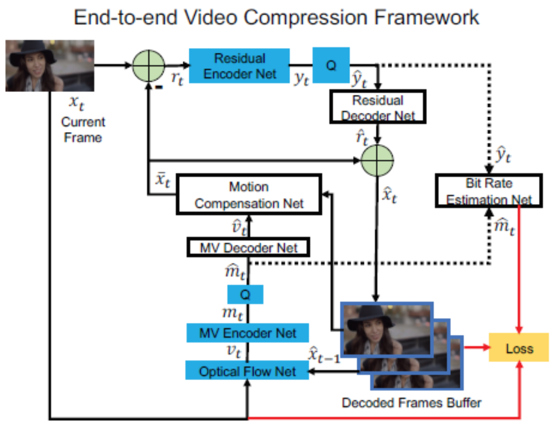


Figure 2: The proposed end-to-end video compression network.

Fig. 2 provides a high-level overview of our end-to-end video compression framework. There is one-to-one correspondence between the traditional video compression framework and our proposed deep learning-based framework. The relationship and brief summarization on the differences are introduced as follows:

Step N1. Motion estimation and compression. We use a CNN model to estimate the optical flow [27], which is considered as motion information v_t. Instead of directly encoding the raw optical flow values, an MV encoder-decoder network is proposed in Fig. 2 to compress and decode the optical flow values, in which the quantized motion representation is denoted as $\hat{m}_t$. Then the corresponding reconstructed motion information $\hat{v}_t$ can be decoded by using the MV decoder net. Details are given in Section 3.3.

Step N2. Motion compensation. A motion compensation network is designed to obtain the predicted frame x_t based on the optical flow obtained in Step N1. More information is provided in Section 3.4.

Step N3-N4. Transform, quantization and inverse transform. We replace the linear transform in Step 3 by using a highly non-linear residual encoder-decoder network, and the residual r_t is non-linearly maped to the representation y_t. Then y_t is quantized to $\hat{y}_t$. In order to build an end-to-end training scheme, we use the quantization method in [11]. The quantized representation $\hat{y}_t$ is fed into the residual decoder network to obtain the reconstructed residual $\hat{r}_t$. Details are presented in Section 3.5 and 3.6.

Step N5. Entropy coding. At the testing stage, the quantized motion representation $\hat{m}_t$ from Step N1 and the residual representation $\hat{y}_t$from Step N3 are coded into bits and sent to the decoder. At the training stage, to estimate the number of bits cost in our proposed approach, we use the CNNs (Bit rate estimation net in Fig. 2) to obtain the probability distribution of each symbol in $\hat{m}_t$ and $\hat{y}_t$. More information is provided in Section 3.6.

Step N6. Frame reconstruction. It is the same as Step 6 in above Section.

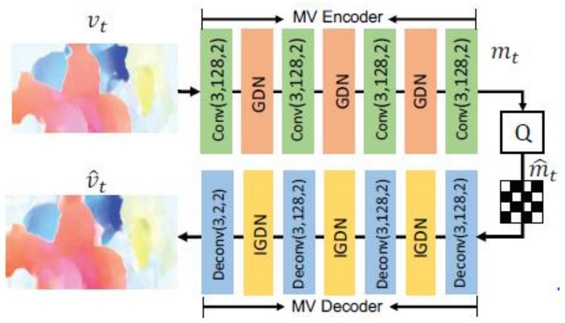


Figure 3: Our MV Encoder-decoder network

In order to compress motion information at Step N1, we design a CNN to transform the optical flow to the corresponding representations for better encoding. Specifically, we utilize an auto-encoder style network to compress the optical flow, which is first proposed by [11] for the image compression task. The whole MV compression network is shown in Fig. 3. The optical flow v_t is fed into a series of convolution operation and nonlinear transform. The number of output channels for convolution (deconvolution) is 128 except for the last deconvolution layer, which is equal to 2. Given optical flow v_t with the size of M × N × 2, the MV encoder will generate the motion representation m_t with the size of M/16×N/16×128. Then m_t is quantized to ˆmt. The MV decoder receives the quantized representation and reconstruct motion information $\hat{v}_t$. In addition, the quantized representation $\hat{m}_t$ will be used for entropy coding.

Given the previous reconstructed frame $\hat{x}_{t-1}$ and the motion vector $\hat{v}_t$, the motion compensation network obtains the predicted frame $\bar{x}_t$, which is expected to as close to the current frame x_t as possible. First, the previous frame ˆx_{t-1} is warped to the current frame based on the motion information $\hat{v}_t$. The warped frame still has artifacts. To remove the artifacts, we concatenate the warped frame w(ˆx_{t-1}, $\hat{v}_t$), the reference frame $\hat{x}_{t-1}$, and the motion vector $\hat{v}_t$ as the input, then feed them into another CNN to obtain the refined predicted frame $\bar{x}_t$. The overall architecture of the proposed network is shown in Fig. 4. The detail of the CNN in Fig. 4 is provided in supplementary material. Our proposed method is a pixel-wise motion compensation approach, which can provide more accurate temporal information and avoid the blockness artifact in the traditional block-based motion compensation method. It means that we do not need the hand-crafted loop filter or the sample adaptive offset technique [39, 31] for post processing.

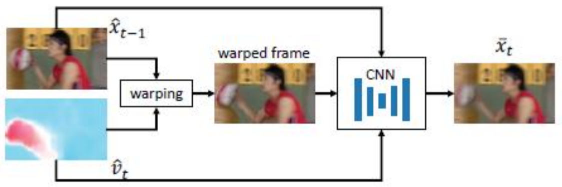


Figure 4: Our Motion Compensation Network.

The residual information rt between the original frame x_t and the predicted frame $\bar{x}_t$ is encoded by the residual encoder network as shown in Fig. 2. In this paper, we rely on the highly non-linear neural network in [12] to transform the residuals to the corresponding latent representation. Compared with discrete cosine transform in the traditional video compression system, our approach can better exploit the power of non-linear transform and achieve higher compression efficiency.

RESULTS AND ANALYSIS

Loss Function. The goal of our video compression framework is to minimize the number of bits used for encoding
the video, while at the same time reduce the distortion between the original input frame x_t and the reconstructed
frame $\hat{x}_t$. Therefore, we propose the following rate-distortion optimization problem,

$$\lambda D + R = \lambda d(x_t, \hat{x}_t) + (H(\hat{m}_t) + H(\hat{y}_t))$$

where $d(x_t, \hat{x}_t)$ denotes the distortion between x_t and $\hat{x}_t$, and we use mean square error (MSE) in our implementation. H(·) represents the number of bits used for encoding the representations. In our approach, both residual representation $\hat{y}_t$ and motion representation $\hat{m}_t$ should be encoded into the bitstreams. λ is the Lagrange multiplier that determines the trade-off between the number of bits and distortion. As shown in Fig. 2, the reconstructed frame $\hat{x}_t$, the original frame x_t and the estimated bits are input to the loss function.

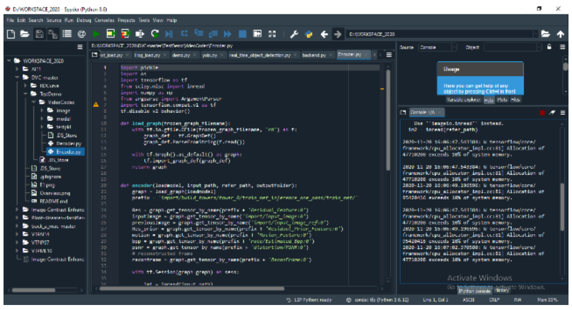
Quantization. Latent representations such as residual representation y_t and motion representation m_t are required to be quantized before entropy coding. However, quantization operation is not differential, which makes end-to-end training impossible. To address this problem, a lot of methods have been proposed [34, 8, 11]. In this paper, we use the method in [11] and replace the quantization operation by adding uniform noise in the training stage. Take y_t as an example, the quantized representation $\hat{y}_t$ in the training stage is approximated by adding uniform noise to y_t, i.e., $\hat{y}_t$

= y_t + η where η is uniform noise. In the inference stage, we use the rounding operation directly, i.e., $\hat{y}_t$ = round (y_t).

Bit Rate Estimation. In order to optimize the whole network for both number of bits and distortion, we need to obtain the bit rate (H($\hat{y}_t$) and H($\hat{m}_t$)) of the generated latent representations ($\hat{y}_t$ and $\hat{m}_t$). The correct measure for bitrate is the entropy of the corresponding latent representation symbols. Therefore, we can estimate the probability distributions of $\hat{y}_t$ and $\hat{m}_t$, and then obtain the corresponding entropy. In this paper, we use the CNNs in [12] to estimate the distributions.

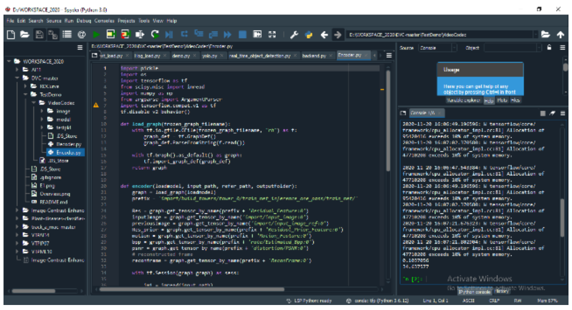
Buffering Previous Frames. As shown in Fig. 2, the previous reconstructed frame $\hat{x}_{t-1}$ is required in the motion estimation and motion compensation network when compressing the current frame. However, the previous reconstructed frame $\hat{x}_{t-1}$ is the output of our network for the previous frame, which is based on the reconstructed frame $\hat{x}_{t-2}$, and so on. Therefore, the frames x_1, . . . , x_{t-1} might be required during the training procedure for the frame x_t, which reduces the variation of training samples in a minibatch and could be impossible to be stored in a GPU when t is large. To solve this problem, we adopt an on-line updating strategy. Specifically, the reconstructed frame $\hat{x}_t$ in each iteration will be saved in a buffer. In the following iterations, $\hat{x}_t$ in the buffer will be used for motion estimation and motion compensation when encoding x_{t+1}. Therefore, each training sample in the buffer will be updated in an epoch. In this way, we can optimize and store one frame for a video clip in each iteration, which is more efficient.

RESULTS



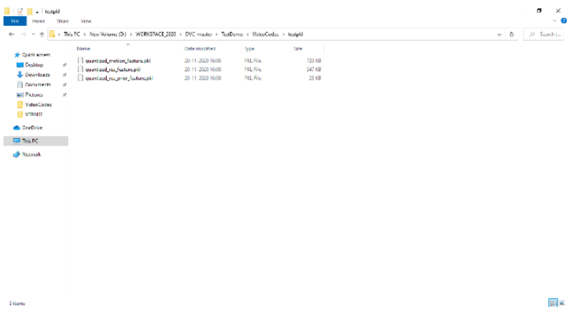
```
import pickle
import os
import tensorflow as tf
from scipy.misc import imread
import numpy as np
from argparse import ArgumentParser
import tensorflow.compat.v1 as tf
tf.disable_v2_behavior()

def load_graph(frozen_graph_filename):
    with tf.io.gfile.GFile(frozen_graph_filename, "rb") as f:
        graph_def = tf.GraphDef()
        graph_def.ParseFromString(f.read())

    with tf.Graph().as_default() as graph:
        tf.import_graph_def(graph_def)
    return graph


def encoder(loadmodel, input_path, refer_path, outputfolder):
    graph = load_graph(loadmodel)
    prefix = 'import/build_towers/tower_0/train_net_inference_one_pass/train_net/'

    Res = graph.get_tensor_by_name(prefix + 'Residual_Feature:0')
    inputImage = graph.get_tensor_by_name('import/input_image:0')
    previousImage = graph.get_tensor_by_name('import/input_image_ref:0')
    Res_prior = graph.get_tensor_by_name(prefix + 'Residual_Prior_Feature:0')
    motion = graph.get_tensor_by_name(prefix + 'Motion_Feature:0')
    bpp = graph.get_tensor_by_name(prefix + 'rate/Estimated_Bpp:0')
    psnr = graph.get_tensor_by_name(prefix + 'distortion/PSNR:0')
    # reconstructed frame
    reconframe = graph.get_tensor_by_name(prefix + 'ReconFrame:0')

    with tf.Session(graph=graph) as sess:
```

Fig 5: Loading Tensorflow



```
import pickle
import os
import tensorflow as tf
from scipy.misc import imread
import numpy as np
from argparse import ArgumentParser
import tensorflow.compat.v1 as tf
tf.disable_v2_behavior()

def load_graph(frozen_graph_filename):
    with tf.io.gfile.GFile(frozen_graph_filename, "rb") as f:
        graph_def = tf.GraphDef()
        graph_def.ParseFromString(f.read())

    with tf.Graph().as_default() as graph:
        tf.import_graph_def(graph_def)
    return graph


def encoder(loadmodel, input_path, refer_path, outputfolder):
    graph = load_graph(loadmodel)
    prefix = 'import/build_towers/tower_0/train_net_inference_one_pass/train_net/'

    Res = graph.get_tensor_by_name(prefix + 'Residual_Feature:0')
    inputImage = graph.get_tensor_by_name('import/input_image:0')
    previousImage = graph.get_tensor_by_name('import/input_image_ref:0')
    Res_prior = graph.get_tensor_by_name(prefix + 'Residual_Prior_Feature:0')
    motion = graph.get_tensor_by_name(prefix + 'Motion_Feature:0')
    bpp = graph.get_tensor_by_name(prefix + 'rate/Estimated_Bpp:0')
    psnr = graph.get_tensor_by_name(prefix + 'distortion/PSNR:0')
    # reconstructed frame
    reconframe = graph.get_tensor_by_name(prefix + 'ReconFrame:0')

    with tf.Session(graph=graph) as sess:

        im1 = imread(input_path)
```

Fig 6: Encoding



This PC › New Volume (D:) › WORKSPACE_2020 › DVC-master › TestDemo › VideoCodec › testpkl

```
Name                              Date modified      Type       Size
quantized_motion_feature.pkl      20-11-2020 16:08   PKL File   720 KB
quantized_res_feature.pkl         20-11-2020 16:08   PKL File   547 KB
quantized_res_prior_feature.pkl   20-11-2020 16:08   PKL File    23 KB
```

Fig 7: Encoded Images (Pkl Files)

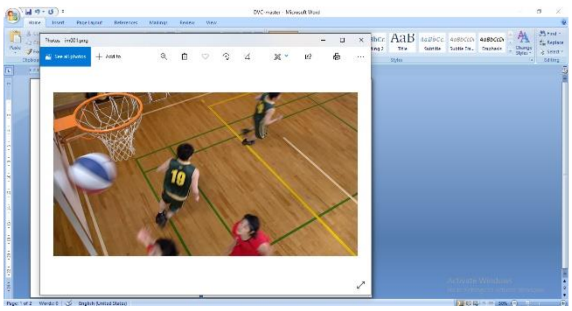


Fig 8: Frame 1

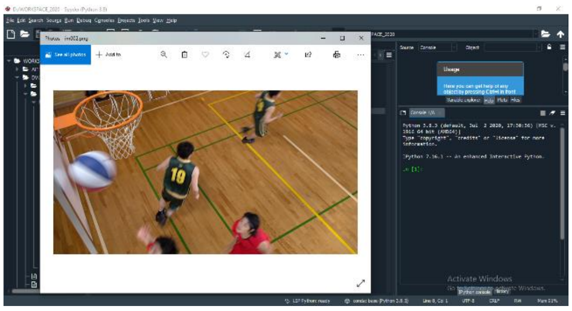


Fig 9: Frame 2

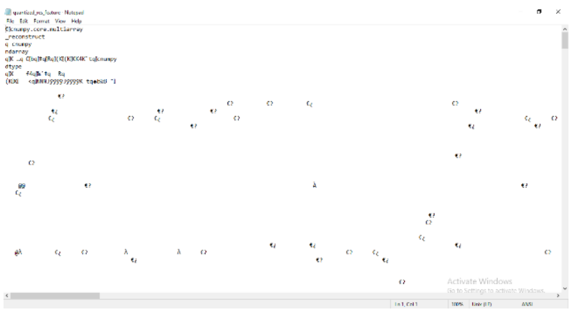


Fig 10: Encoded Frame 1

Fig 11: Encoded Frame 2

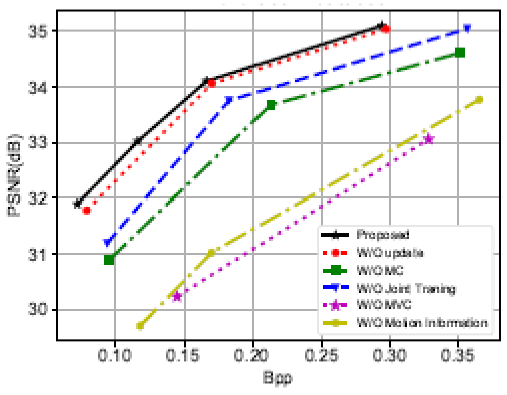


Fig 11: Encoded Frame 2

CONCLUSION

In this paper, we have proposed the fully end-to-end deep learning framework for video compression. Our framework inherits the advantages of both classic predictive coding scheme in the traditional video compression standards and the powerful non-linear representation ability from DNNs. Experimental results show that our approach outperforms the widely used H.264 video compression standard and the recent learning-based video compression system. The work provides a promising framework for applying deep neural network for video compression. Based on the proposed framework, other new techniques for optical flow, image compression, bi-directional prediction and rate control can be readily plugged into this framework.

REFERENCES

[1] F. bellard, bpg image format. http://bellard.org/bpg/. Accessed: 2018-10-30.

[2] The h.264/avc reference software. http://iphome.hhi.de/suehring/. Accessed: 2018-10-30.

[3] Hevc test model (hm). https://hevc.hhi.fraunhofer.de/HM-doc/. Accessed: 2018-10-30.

[4] Ultra video group test sequences. http://ultravideo.cs.tut.fi. Accessed: 2018-10-30.

[5] Webp. https://developers.google.com/speed/webp/. Accessed: 2018-10-30.

[6] x264, the best h.264/avc encoder. https://www.videolan.org/developers/x264.html.Accessed: 2018-10-30.

[7] x265 hevc encoder / h.265 video codec. http://x265.org. Accessed: 2018-10-30.

[8] E. Agustsson, F. Mentzer, M. Tschannen, L. Cavigelli, R. Timofte, L. Benini, and L. V. Gool. Soft-to-hard vector quantization for end-to-end learning compressible representations. In NIPS, pages 1141–1151, 2017.

[9] E. Agustsson, M. Tschannen, F. Mentzer, R. Timofte, and L. Van Gool. Generative adversarial networks for extreme learned image compression. arXiv preprint arXiv:1804.02958, 2018.

[10] M. H. Baig, V. Koltun, and L. Torresani. Learning to inpaint for image compression. In NIPS, pages 1246–1255, 2017.

[11] J. Ball´e, V. Laparra, and E. P. Simoncelli. End-to-end optimized image compression. arXiv preprint arXiv:1611.01704, 2016.

[12] J. Ball´e, D. Minnen, S. Singh, S. J. Hwang, and N. Johnston. Variational image compression with a scale hyperprior. arXiv preprint arXiv:1802.01436, 2018.

[13] T. Chen, H. Liu, Q. Shen, T. Yue, X. Cao, and Z. Ma. Deepcoder: A deep neural network-based video compression. In VCIP, pages 1–4. IEEE, 2017.

[14] Z. Chen, T. He, X. Jin, and F. Wu. Learning for video compression. arXiv preprint arXiv:1804.09869, 2018.

[15] A. Dosovitskiy, P. Fischer, E. Ilg, P. Hausser, C. Hazirbas, V. Golkov, P. Van Der Smagt, D. Cremers, and T. Brox. Flownet: Learning optical flow with convolutional networks. In ICCV, pages 2758–2766, 2015.

[16] 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.

[17] T.-W. Hui, X. Tang, and C. Change Loy. Lite flow net: A lightweight convolutional neural network for optical flow estimation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 8981–8989, 2018.

[18] T.-W. Hui, X. Tang, and C. C. Loy. A lightweight optical flow cnn–revisiting data fidelity and regularization. arXiv preprint arXiv:1903.07414, 2019.

[19] N. Johnston, D. Vincent, D. Minnen, M. Covell, S. Singh, T. Chinen, S. Jin Hwang, J. Shor, and G. Toderici. Improved lossy image compression with priming and spatially adaptive bit rates for recurrent networks. In CVPR, June 2018.

[20] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980, 2014.

[21] M. Li,W. Zuo, S. Gu, D. Zhao, and D. Zhang. Learning convolutional networks for content-weighted image compression. In CVPR, June 2018.

[22] Z. Liu, X. Yu, Y. Gao, S. Chen, X. Ji, and D. Wang. Cu partition mode decision for hevc hardwired intra encoder using convolution neural network. TIP, 25(11):5088–5103, 2016.

[23] G. Lu,W. Ouyang, D. Xu, X. Zhang, Z. Gao, and M.-T. Sun. Deep kalman filtering network for video compression artifact reduction. In ECCV, September 2018.

[24] F. Mentzer, E. Agustsson, M. Tschannen, R. Timofte, and L. Van Gool. Conditional probability models for deep image compression. In CVPR, number 2, page 3, 2018.

[25] D. Minnen, G. Toderici, M. Covell, T. Chinen, N. Johnston, J. Shor, S. J. Hwang, D. Vincent, and S. Singh. Spatially adaptive image compression using a tiled deep network. In ICIP, pages 2796–2800. IEEE, 2017.

[26] C. V. networking Index. Forecast and methodology, 2016-2021, white paper. San Jose, CA, USA, 1, 2016.

[27] A. Ranjan and M. J. Black. Optical flow estimation using a spatial pyramid network. In CVPR, volume 2, page 2. IEEE, 2017.

[28] O. Rippel and L. Bourdev. Real-time adaptive image compression. In ICML, 2017.

[29] A. Skodras, C. Christopoulos, and T. Ebrahimi. The jpeg 2000 still image compression standard. IEEE Signal Processing Magazine, 18(5):36–58, 2001.

[30] R. Song, D. Liu, H. Li, and F. Wu. Neural network-based arithmetic coding of intra prediction modes in hevc. In VCIP, pages 1–4. IEEE, 2017.

[31] G. J. Sullivan, J.-R. Ohm, W.-J. Han, T. Wiegand, et al. Overview of the high efficiency video coding(hevc) standard. TCSVT, 22(12):1649–1668, 2012.

[32] D. Sun, X. Yang, M.-Y. Liu, and J. Kautz. PWC-net: CNNS for optical flow using pyramid, warping, and cost volume. In CVPR, pages 8934–8943, 2018.

[33] L. Theis, W. Shi, A. Cunningham, and F. Husz´ar. Lossy image compression with compressive autoencoders. arXiv preprint arXiv:1703.00395, 2017.

[34] G. Toderici, S. M. O'Malley, S. J. Hwang, D. Vincent, D. Minnen, S. Baluja, M. Covell, and R. Sukthankar. Variable rate image compression with recurrent neural networks. arXiv preprint arXiv:1511.06085, 2015.

[35] G. Toderici, D. Vincent, N. Johnston, S. J. Hwang, D. Minnen, J. Shor, and M. Covell. Full resolution image compression with recurrent neural networks. In CVPR, pages 5435–5443, 2017.

[36] Y.-H. Tsai, M.-Y. Liu, D. Sun, M.-H. Yang, and J. Kautz. Learning binary residual representations for domain-specific video streaming. In Thirty-Second AAAI Conference on Artificial Intelligence, 2018.

[37] G. K. Wallace. The jpeg still picture compression standard. IEEE Transactions on Consumer Electronics, 38(1): xviii– xxxiv, 1992.

[38] Z. Wang, E. Simoncelli, A. Bovik, et al. Multi-scale structural similarity for image quality assessment. In ASILOMAR CONFERENCE ON SIGNALS SYSTEMS AND COMPUTERS, volume 2, pages 1398–1402. IEEE; 1998, 2003.

[39] T. Wiegand, G. J. Sullivan, G. Bjontegaard, and A. Luthra. Overview of the h. 264/avc video coding standard. TCSVT, 13(7):560–576, 2003.

[40] C.-Y. Wu, N. Singhal, and P. Krahenbuhl. Video compression through image interpolation. In ECCV, September 2018.

[41] C.-Y. Wu, M. Zaheer, H. Hu, R. Manmatha, A. J. Smola, and P. Kr¨ahenb¨uhl. Compressed video action recognition. In CVPR, pages 6026–6035, 2018.

[42] T. Xue, B. Chen, J. Wu, D. Wei, and W. T. Freeman. Video enhancement with task-oriented flow. arXiv preprint arXiv:1711.09078, 2017.