Learned Video Compression with Feature-level Residuals

Runsen Feng, Yaojun Wu, Zongyu Guo, Zhizheng Zhang, Zhibo Chen*
CAS Key Laboratory of Technology in Geo-spatial Information Processing and Application System
University of Science and Technology of China
{fengruns,yaojunwu,guozy,zhizheng}@mail.ustc.edu.cn, chenzhibo@ustc.edu.cn

Abstract

In this paper, we present an end-to-end video compression network for P-frame challenge on CLIC. We focus on deep neural network (DNN) based video compression, and improve the current frameworks from three aspects. First, we notice that pixel space residuals is sensitive to the prediction errors of optical flow based motion compensation. To suppress the relative influence, we propose to compress the residuals of image feature rather than the residuals of image pixels. Furthermore, we combine the advantages of both pixel-level and feature-level residual compression methods by model ensembling. Finally, we propose a step-by-step training strategy to improve the training efficiency of the whole framework. Experiment results indicate that our proposed method achieves 0.9968 MS-SSIM on CLIC validation set and 0.9967 MS-SSIM on test set.

1. Introduction

Video data has occupied more than 80% internet transmission resources [5] in recent years, and the percentage will be increased even further. As the demand for video transmission grows, traditional video codec has been studied for decades as a way to save internet bandwidth. It typically tackles the compression problem with four basic steps: prediction, transform, quantization and entropy coding. Although it has been developed for many years, it still updates a new generation every 8 to 10 years, and each generation can bring nearly 50% gain on bandwidth saving.

Recently, deep learning technology opens countless possibilities in further improving the coding performance [4, 12, 9, 4, 7, 6]. Unlike traditional video codec, DNN based methods can jointly optimize their network components in an end-to-end manner, which can further improve the rate-distortion performance of compression [9]. Among DNN based methods, Chen *et al.* [3] first propose to predict block of pixels autoregressively, and the residuals is encoded by an autoencoder. Besides, Wu *et al.* [12] propose a interpolation-based approach with traditional motion vectors. They recurrently compress the residuals conditioned on warped reference frames and context. Following conventional video coding architecture, Lu *et al.* [9] propose an end-to-end trainable framework by replacing all key components in the classical video codec with DNNs. In their framework, optical flow is used for motion compensation, and two autoencoders are adopted to encode the corresponding optical flow and residuals, respectively. [6] perform temporal interpolation by the decoded optical flow and blending coefficients, and directly quantize the latent space residuals by reusing the same autoencoder of image compression. Besides, Habibian *et al.* [7] propose a rate-distortion autoencoder which consists of a 3D autoencoder transform and an autoregressive model for entropy coding in latent space.

In this paper, we propose an end-to-end video compression framework based on [9] for CLIC 2020 P-frame task. We focus on improving the performance of residual coding, which is not fully explored in previous methods. Our contributions can be summarized as follows:

- We compress the residual information from the perspective of feature level instead of pixel level, which can effectively suppress the influence of motion compensation error.
- We involve both of our pixel-level and feature-level residual compression methods within an ensemble model, which further improves the compression performance.
- To reduce the difficulty of training multiple modules from scratch, we thus design a step-by-step training strategy to improve the training efficiency.

2. Method

2.1. Overall Framework

In this section, we give a high-level overview of our framework. Figure 1 shows the overall coding pipeline as

*Corresponding author.

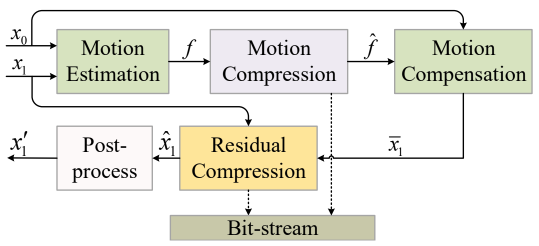


Figure 1: The overall framework of our proposed method.

well as basic components. Brief summarization on these components is introduced as follows:

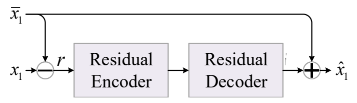
Motion Estimation. We apply PWC-Net [11] as our motion estimation network to compute optical flow between two adjacent frames. Specifically, the target frame $\boldsymbol{x}_1$ and the reference frame $\boldsymbol{x}_0$ are fed into the motion estimation network to extract motion information $\boldsymbol{f}$, which will be encoded by motion compression module.

Motion Compression. To compress optical flow for transmission, we propose a VAE based motion compression network based on the factorized-prior model proposed by [2]. We replace GDN with residual blocks and further add a downsampling layer. This is because the spatial redundancy in optical flow is much higher than that in image. Through motion compression, we can obtain reconstructed optical flow $\hat{\boldsymbol{f}}$ for motion compensation.

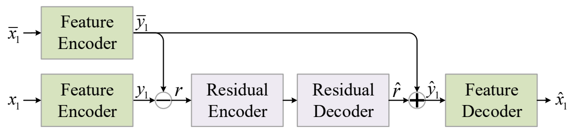
Motion Compensation. In motion compensation, we try to obtain prediction frame $\overline{\boldsymbol{x}}_1$ through the decoded optical flow $\hat{\boldsymbol{f}}$ and the reference frame $\boldsymbol{x}_0$. Detailed structure of the motion compensation network can be seen in [9].

Residual Compression. We aim to encode the residual information between the compensation frame $\overline{\boldsymbol{x}}_1$ and the original frame $\boldsymbol{x}_1$. There are two variants of our residual compression networks as shown in 2. For the first variant, we directly compute the pixel space residuals which then are compressed as image data. We employ our pixel-level residual compression network based on the hyper-prior model in [2]. For the second variant, we calculate and compress residuals in feature domain. The corresponding feature-level residual compression network will be described in section 2.2.

Post-process Inspired by the success of image quality enhancement networks, we add GRDN [8] as our post-process network to enhance the quality of reconstructed frame $\hat{\boldsymbol{x}}_1$ after residual compression. The final output of the post-process network is denoted as $\boldsymbol{x}_1'$.



(a) Residual compression on pixel level.



(b) Residual compression on feature level.

Figure 2: Two residual compression methods in our framework.

2.2. Feature-level Residuals

Residual compression module plays an important role in exploring the remaining spatial redundancy conditioned on motion compensation frame. Traditional approaches typically calculate and compress the residuals in pixel space. However, the corresponding residuals are easily affected by the prediction error from optical flow based motion compensation. Instead of directly computing pixel-level residuals, we propose to compress the residuals of image feature. As shown in Figure 2(b), the compensation frame $\overline{\boldsymbol{x}}_1$ and the target frame $\boldsymbol{x}_1$ are transformed into feature space by a shared feature encoder, and then the residual of their feature is compressed by a variational autoencoder. Note that different from the work [6] which directly compute and quantize latent space residuals to reuse image compression network, our method calculate residuals in high-dimensional feature space and further transform the residuals into a discrete latent representation.

The detailed structure of the proposed residual compression networks is illustrated in Figure 3. Similar to exiting work [2] for learned image compression, the VAE for residual compression consists of an autoencoder transform and a hierarchical entropy model. The autoencoder first transforms the feature-level residual $\boldsymbol{r}$ into a quantized latent representation $\hat{\boldsymbol{m}}$ and then maps it back to reconstruction $\hat{\boldsymbol{r}}$. A non-parametric, fully factorized density model proposed by Ballé *et al.* [2] is used for estimating the distribution of the hyperprior $\hat{\boldsymbol{z}}$. Conditioned on $\hat{\boldsymbol{z}}$, the distribution of each latent $\hat{m}_i$ is modeled as a Gaussian Mixture convolved with a unit uniform distribution:

$$p_{\hat{\boldsymbol{m}}|\hat{\boldsymbol{z}}}(\hat{\boldsymbol{m}} \mid \hat{\boldsymbol{z}}) = \prod_i (\sum_{k=1}^{K} \pi_{i,k}\mathcal{N}(\mu_{i,k}, \sigma_{i,k}^2) * \mathcal{U}(-\frac{1}{2}, \frac{1}{2}))(\hat{m}_i) \quad (1)$$

where k denotes the index of mixtures. We use $K = 3$ in our experiments. The parameters of weight, mean and scale are predicted by the parameter network. Following the work of [1], we replace the rounding function Q (quantization) with additive uniform noise during training.

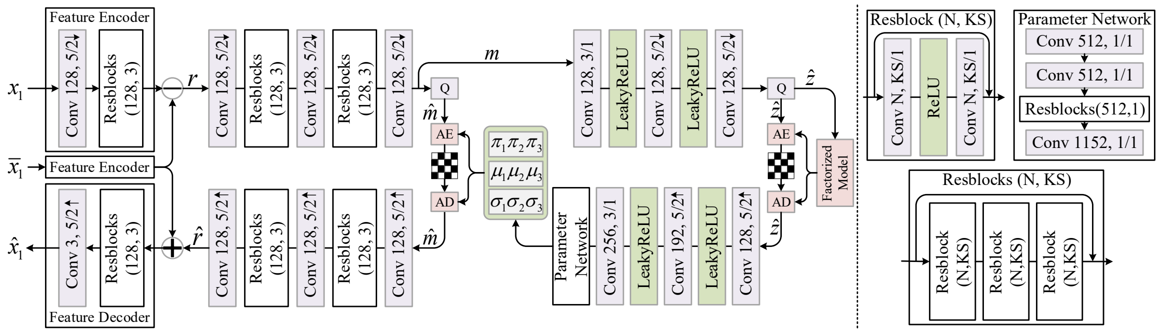


Figure 3: Framework of residual compression in feature space. Q represents quantization and AE, AD respectively represent arithmetic encoder and arithmetic decoder. The parameters of convolution are denoted as numbers of filters, kernel size/down- or upsampling stride, where ↓ indicates down sampling and ↑ indicates up sampling.

2.3. Model Ensemble

When motion compensation is accurate, feature-level residuals have no more advantage than pixel-level residuals. To further improve the compression performance, we train two individual P-frame frameworks which contain the two residual compression networks shown in Figure 2, respectively. When encoding the validation dataset, both of the frameworks are used to compress each sequence and then an optimal assignment is chosen by a knapsack solver given the target bitrate. Experimental results show that the ensemble of these two frameworks outperforms any of them.

2.4. Training Strategy

Loss Function. The overall rate-distortion (R-D) loss function can be fomulated as follows:

$$\mathcal{L} = R_f + R_r + \lambda d(\boldsymbol{x}_1, \boldsymbol{x}_1') \tag{2}$$

where R_f and R_r represent the rate of optical flow and residual, respectively. The distortion d is measured using MS-SSIM and the coefficient λ controls the R-D trade-off.

Step-by-step Training. It is hard to train the whole framework from scratch. Therefore, we divide the training procedure into pre-training stage and fine-tuning stage. In the pre-training stage, we first train the motion compression and motion compensation networks, keeping the parameters of the pre-trained PWC-Net in [11] fixed. Then the residual compression networks are added for joint training with a large λ. This is to avoid information loss in pre-training stage. The post-process network is pre-trained on our learned image codec. In the second stage, we jointly fine-tune all of the modules with different λ.

3. Experiments

3.1. Implementation Details

We train our P-frame compression model with two datasets provided by Vimeo-90K[13] and CLIC, respectively. The Vimeo-90k dataset consists of 89800 clips in an RGB format and the CLIC dataset contains about 466684 frames of YUV420 format. The YUV420 images in the CLIC dataset are converted into YUV444. Each pair of frames are randomly cropped into 256x256 patches during training. The mini-batch size is set as 8.

In the pre-training stage, we train our model on the Vimeo-90k dataset using the learing rate of 5×10^{-5}. In the fine-tuning stage, we set learning rate to 1×10^{-5} and jointly fine-tune the whole framework on the CLIC dataset. We convert $\boldsymbol{x}_1$ and $\boldsymbol{x}_1'$ back into YUV420 format to calculate distortion when fine-tuning. It takes about 7 days to complete the training procedure using two GTX 1080 Ti GPUs.

The total size in P-frame task, which consists of the compressed data size and model size, should be no more than a target size. Thus, we train our framework with different λ and select two most promising models to perform bit-allocation by a knapsack solver.

3.2. Experimental Results

We evaluate several design choices on the CLIC validation dataset. Specifically, we evaluate the benefits of the post-process module, the proposed feature-level residuals and the multi-model ensemble. For a fair comparison, two models trained by different λ are used to make bit-allocation for each framework .

Evaluation results are shown in Table 1. It is obvious that the proposed feature-level residuals consistently bring performance gain, with or without post-process networks. When GRDN is included, significant performance improvement occurs on the framework of pixel-level residuals. Compared with GRDN, the results in second and third rows indicate that our feature-level residual compression module performs better with less model parameters. The ensemble modeling further improves the performance as shown in the fifth and sixth rows of the table. Due to time reason, we only submit the version which achieves

Table 1: Evaluation results on P-frame validation dataset.

Model #	Pixel-level residuals	Feature-level residuals	GRDN	Ensemble	Data size	Model size	MS-SSIM
# 1	✓				38205911	79114205	0.996302
# 2	✓		✓		37788576	120847836	0.996619
# 3		✓			38133059	86399558	0.996645
# 4		✓	✓		37716003	128105152	0.996700
# 5	✓	✓	✓	# 1, # 4	37960950	103610323	0.996792
# 6	✓	✓	✓	# 2, # 4	37735411	126164272	0.996866

* The unit of data/model size is Byte. For P-frame challenge in CLIC, we limit the total size to 3,900,000,000 bytes, which is calculated as follows: model size + 100 $\times$ data size.

0.996792 MS-SSIM on validation dataset. Our final ensemble model, which reaches 0.996866 MS-SSIM, is comprised of the models in the second and fourth rows.

To satisfied the limitation of decoding time, the autoregressive entropy model in [10] is not implemented in our framework. The introduction of autoregressive models can further boost the performance for both motion compression and residual compression modules.

4. Conclusion

In this paper, we propose an end-to-end video compression framework for the P-frame challenge in CLIC 2020. Firstly, we propose a feature-level residual compression network to reducing the effects of motion compensation error. Secondly, we combine two different residual compression frameworks in an ensemble model to further improve the performance. Lastly, a step-by-step training strategy is proposed to further improve the training efficiency. As shown in the leaderboard of P-frame validation, our team 'IMCL_MSSSIM' achieved 0.9968 MS-SSIM score. In the future, finding an end-to-end structure to jointly exploit both pixel-level and feature-level residuals would be an interesting direction.

Acknowledgment

This work was supported in part by NSFC under Grant U1908209, 61632001 and the National Key Research and Development Program of China 2018AAA0101400.

References

[1] Johannes Ballé, Valero Laparra, and Eero P Simoncelli. End-to-end optimized image compression. *arXiv preprint arXiv:1611.01704*, 2016. 2

[2] Johannes Ballé, David Minnen, Saurabh Singh, Sung Jin Hwang, and Nick Johnston. Variational image compression with a scale hyperprior. *arXiv preprint arXiv:1802.01436*, 2018. 2

[3] Zhibo Chen, Tianyu He, Xin Jin, and Feng Wu. Learning for video compression. *IEEE Transactions on Circuits and Systems for Video Technology*, 30(2):566–576, 2020. 1

[4] Zhengxue Cheng, Heming Sun, Masaru Takeuchi, and Jiro Katto. Learning image and video compression through spatial-temporal energy compaction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 10071–10080, 2019. 1

[5] VNI Cisco. Cisco visual networking index: Forecast and trends, 2017–2022. *White Paper*, 1, 2018. 1

[6] Abdelaziz Djelouah, Joaquim Campos, Simone Schaub-Meyer, and Christopher Schroers. Neural inter-frame compression for video coding. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 6421–6429, 2019. 1, 2

[7] Amirhossein Habibian, Ties van Rozendaal, Jakub M Tomczak, and Taco S Cohen. Video compression with rate-distortion autoencoders. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 7033–7042, 2019. 1

[8] Dong-Wook Kim, Jae Ryun Chung, and Seung-Won Jung. Grdn: Grouped residual dense network for real image denoising and gan-based real-world noise modeling. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 0–0, 2019. 2

[9] Guo Lu, Wanli Ouyang, Dong Xu, Xiaoyun Zhang, Chunlei Cai, and Zhiyong Gao. Dvc: An end-to-end deep video compression framework. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 11006–11015, 2019. 1, 2

[10] David Minnen, Johannes Ballé, and George D Toderici. Joint autoregressive and hierarchical priors for learned image compression. In *Advances in Neural Information Processing Systems*, pages 10771–10780, 2018. 4

[11] Deqing Sun, Xiaodong Yang, Ming-Yu Liu, and Jan Kautz. Pwc-net: Cnns for optical flow using pyramid, warping, and cost volume. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8934–8943, 2018. 2, 3

[12] Chao-Yuan Wu, Nayan Singhal, and Philipp Krahenbuhl. Video compression through image interpolation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 416–431, 2018. 1

[13] Tianfan Xue, Baian Chen, Jiajun Wu, Donglai Wei, and William T Freeman. Video enhancement with task-oriented flow. *International Journal of Computer Vision*, 127(8):1106–1125, 2019. 3