DeepCoder: A Deep Neural Network Based Video Compression

Tong Chen[†], Haojie Liu[†], Qiu Shen, Tao Yue, Xun Cao, Zhan Ma
School of Electronic Science and Engineering, Nanjing University, Jiangsu, China
Emails: {tong, haojie}@smail.nju.edu.cn, {shenqiu, yuetao, caoxun, mazhan}@nju.edu.cn
[†]*Authors contributed equally.*

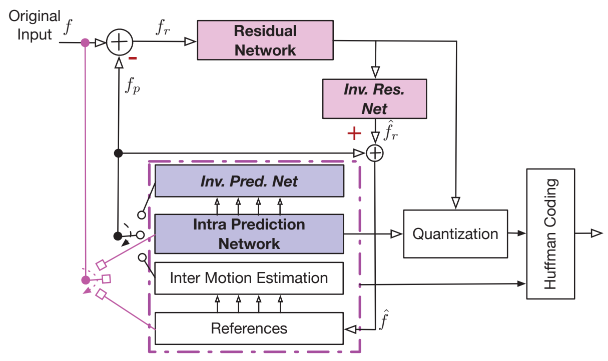
***Abstract*—Inspired by recent advances in deep learning, we present the DeepCoder - a Convolutional Neural Network (CNN) based video compression framework. We apply separate CNN nets for predictive and residual signals respectively. Scalar quantization and Huffman coding are employed to encode the quantized feature maps (fMaps) into binary stream. We use the fixed 32×32 block in this work to demonstrate our ideas, and performance comparison is conducted with the well-known H.264/AVC video coding standard with comparable rate-distortion performance. Here distortion is measured using Structural Similarity (SSIM) because it is more close to perceptual response.**

***Index Terms*—Deep neural network, video compression, prediction, residual**

I. Introduction

Video compression is a vital process for local storage and streaming over the network. For instance, it is about 93 mega bytes (MB) for a second 1080p 30 Hz (i.e., fps or frame per second) raw video content using YUV 420 format. None of recent wireless networks could sustain this raw content with such high data rate in real-time. Almost all famous video compression standards are based on the hybrid motion-compensation and transform coding framework [1]. Predictive signals f_p are constructed using neighbor pixels or temporal displaced blocks. Residuals f_r are then transformed, quantized and finally entropy coded. Necessary filters are placed, such as deblocking, sample adaptive offset (SAO), to improve the quality of reconstructed signals $\hat{f}$.

Meanwhile, researchers have attempted to explore other possibilities, such as using learned dictionaries, either with fixed or self-adaptive atoms [2]–[4], plus fixed transform basis (e.g., DCT). With such implementation, blocks are represented by weighted (sparse) dictionaries and transformed basis coefficients.



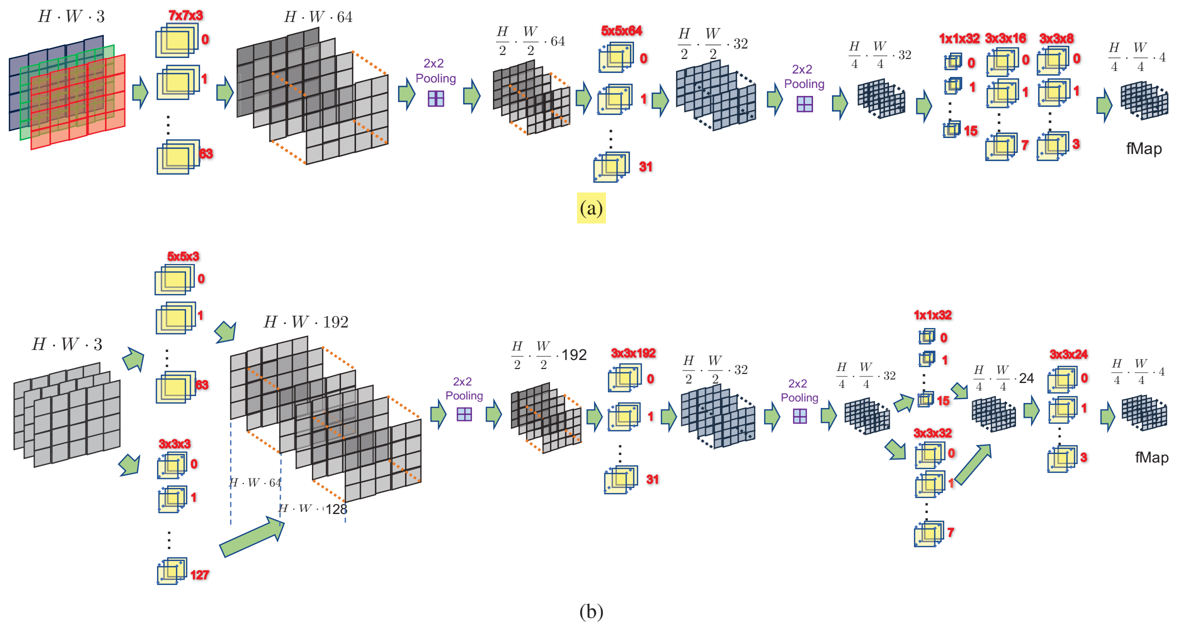
(a)

Fig. 1. System architecture of our deep neural network based video encoder

Inspired by recent advances in deep learning, neural network based image lossy compression [5]–[7] attracts the attentions from both academia and industry. Particularly, Google researchers recently have published two papers concerning the thumbnail and full resolution image compression [6], [7] using recurrent neural networks (RNN). It consists of concatenated (stacked) autoencoders that use trained network to represent blocks with pre-defined bit depth (i.e., 4-bit as exemplified). Residuals between input block and representation using trained network basis will be fed into next stage using the same trained network. This is so called recurrent fashion. Corresponding coefficients at each step will be encoded. The more steps, the more bits required for block representation and the better reconstructed quality. As we can see, signal block that is using trained network basis at each step, is a kind of decomposition using adaptive transform basis.

Different from all above existing techniques in literature, we have proposed a new framework for lossy video coding, leveraging the advantages from both "learned dictionary" and "neural network". Given the successful history of video compression using the classical ideas of "prediction" plus "transform", we treat any input video signal as the combination of the "predictive" and "error (residual)" samples, i.e., $f = f_p + f_r$. We propose to use the trained neural networks to represent these two parts, with corresponding coefficients coded into the stream. Independent networks are applied to the "prediction" and "residual", given the quite distant mathematical signal distribution respectively. Scalar quantization is used to encode coefficients followed by the most straightforward Huffman entropy coding. For simplicity, we have applied the processing using the fixed 32×32 block for both intra and inter coding in this work. Only one reference frame is used for temporal prediction.

Compared with the famous H.264/AVC, our DeepCoder has shown similar rate-distortion efficiency when performing tests using test sequences commonly adopted in video coding standardization community, with about 1% BD-Rate loss. Here BD-Rate is measured using SSIM [8] and Bit Rate. Overall, as for such preliminary exploration using the neural networks



$H \cdot W \cdot 3$
7x7x3
$H \cdot W \cdot 64$
2x2 Pooling
$\frac{H}{2} \cdot \frac{W}{2} \cdot 64$
5x5x64
$\frac{H}{2} \cdot \frac{W}{2} \cdot 32$
$\frac{H}{4} \cdot \frac{W}{4} \cdot 32$
1x1x32
3x3x16
3x3x8
$\frac{H}{4} \cdot \frac{W}{4} \cdot 4$
fMap
(a)
5x5x3
3x3x3
$H \cdot W \cdot 192$
$H \cdot W \cdot 64$
$H \cdot W \cdot 128$
$\frac{H}{2} \cdot \frac{W}{2} \cdot 192$
3x3x192
$\frac{H}{4} \cdot \frac{W}{4} \cdot 24$
3x3x32
3x3x24
(b)

Fig. 2. Network architecture (i.e., layers, convolutional kernel parameters) of our proposed image coding system (a) prediction; (b) residual.

for video coding, our DeepCoder shows the potential and we believe the coding efficiency will be significantly improved with future fine tuning and careful coding tool design.

The rest of this work is organized as follows: Section II details the system architecture that we propose to use neural network for video compression followed by the performance evaluation in Section III. Finally, concluding remarks are drawn in Section IV.

II. System Architecture of Neural Network Based Video Compression

This section introduces the system architecture of our deep video coder as shown in Fig. 1. Either intra- or inter- correlations are utilized to form compactly represented predictions of image blocks through the predictive encoder, and residuals are compressed by using the inter-\intra- residual network. Both predictive and residual coefficients are quantized and entropy coded to produce the final binary stream. As shown in Fig. 1, the entire coding system including prediction, residual coding, quantization and entropy coding. Filtering process that is applied in existing video standards, such as deblocking, SAO, is not considered in this work. In our system, three convolution neural networks are used for predicting the blocks from intra information and compressing both inter and intra residuals. Residuals share the identical network architecture but different parameters.

A. Predictive Encoder

Prediction is the most important module in our deep video coding system. We introduce the intra-/inter- predictions as the two main prediction methods to generate the predictive signals. At this step, each block of current frame is predicted from either the former frame (i.e. inter-prediction) or intra information (i.e. intra-prediction). The intra-/inter- selection is applied to select the better prediction method.

a) Intra-\inter- selection.: Specifically, all the frames of the input video are split into non-overlapped 32×32 image blocks. The mean square error (MSE) of the inter-predicted block is computed. If the MSE is smaller than a certain constant threshold, the inter-prediction is applied, on the contrary, if the MSE is larger than the threshold, the intra-prediction is preferred.

b) Inter-prediction and motion estimation.: The inter-prediction predicts image blocks by using similar patches from the former frame. A brute force search strategy is applied to offset the motion induced translation and find the best matching block. Specifically, we applied a moving 32×32 window to pick out patches in a local area of the former frames at the inter-prediction candidates, and the best matching block is selected by choosing one of these candidates with minimal MSE. As for the motion search range, a 21×21 searching window is used to balance the performance and complexity as well.

Furthermore, we introduce another SKIP mode as a special inter case when block MSE is less than a threshold. SKIP mode has proven to be very effective for stationary areas such as background. This mode is inferred using a binary flag in the stream. For each block marked as the SKIP mode, we only stream its motion vector and force its residual to zero.

c) Intra-prediction network.: The intra-prediction try to predict image blocks by using spatial correlations contained in themselves. Inspired by the recent successes of deep learning, we design a intra-prediction network in this paper, and present its promising performance in video coding tasks. We apply a five-layer convolutional network as the encoder, as shown in Fig. 2(a). Decoder is the symmetric counterpart as the

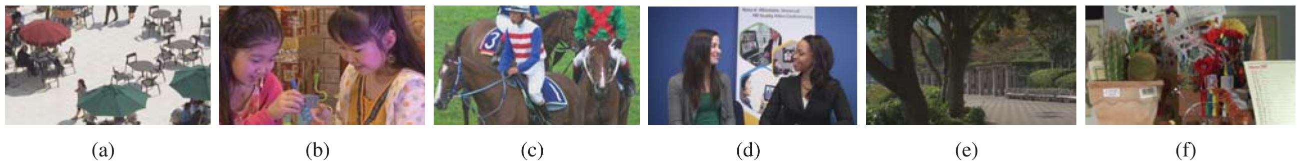


Fig. 3. Sample videos for performance evaluation (a) BQSquare (b) BasketballPass (c) RaceHorses (d) KristenAndSara (e) ParkScene (f) Cactus

encoder. For a $H \times W \times 3$ RGB image, final output is a feature map (fMap) at a reduced size of $\frac{H}{4} \times \frac{W}{4} \times 4$. This fMap is further quantized and encoded into the stream. Reconstruction is applied using the symmetric decoder network to derive the intra prediction signal (but with quantization noise).

B. Residual Encoder

After prediction, input blocks are represented by either similar patches in former frames or compact fMap. However, both of them lead to quality deterioration caused by prediction residuals. To better reconstruct videos, we introduce the residual coding to handle both residuals of intra-prediction and inter-prediction. To simplify the problem, we use the same network structures as shown in Fig. 2(b) for both residuals, but may use different convolutional weights. To capture the features efficiently in residual domain, we apply the inception architecture [9] to combine the outputs of two parallel convolutional process.

C. Quantization and Entropy Coding

In this paper, a scalar quantization and Huffman entropy coding method are applied to further compress the final output fMaps of both intra prediction and residuals.

First, we scale the floating coefficients to 8-bit integer. We then apply the simplest quantization by reducing the bit depth from the most significant bit (Msb) to the least significant bit (Lsb) directly, resulting in $8 \times (8 + 1) = 72$ combinations (assuming zero residuals) of reconstructed signal with respect to the pair of rate and distortion measurement. The one producing best rate-distortion (R-D) performance is selected. Here, the best R-D point locates at the top of the R-D curves.

After the quantization, the Huffman code is utilized. The table of variable-length code is derived according to the histogram of quantized fMaps. The short codes are assigned to the symbols with high probability of occurrence, while the longer ones are used to represent the rarely occurred symbols. Similar process is extended to the motion vector coding as well.

D. Network Training

Here, we introduce more details on how to train the convolutional neural networks used in the work. In practice, we use the Twitter Image Dataset mentioned in the AR-CNN [10] for training. All the 500 images are split into 32x32 patches, where 75% (660000 patches) of them are used as the training set and the rest (165000 patches) is the cross-validation set. Since all the encoding networks involved in this paper has their counterpart decoding networks, we train the encoder

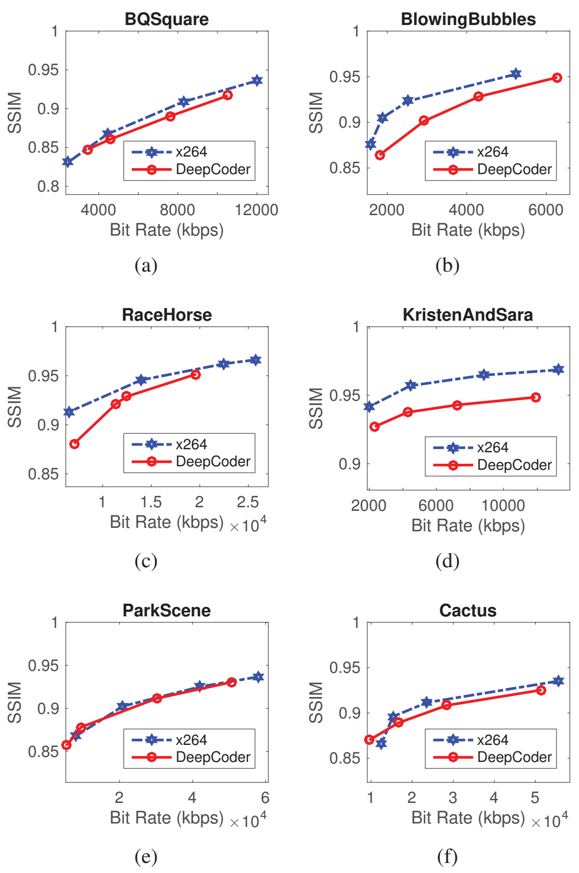


Fig. 4. SSIM vs. Bit Rate (kbps) for our DeepCoder and x264.

and decoder in pair-wise. The original image X_n regarded as the input is compressed by CNN encoder and output the reconstructed image Y_n by the associated CNN decoder. The objective of training is to minimize the following loss function $L(\theta) = \frac{1}{N}\sum_{n=1}^{N} ||Y_n(\theta) - X_n||^2$, where Y_n is reconstructed imageX_n is input image, and N is batch size.

We use the optimizer adadelta (an adaptive learning rate method) [11] to make fast convergence and generate the pre-training model first after 100 epochs. To avoid getting the local optimum, we turn to using another optimizer adam (a method for stochastic optimization) [12] based on the pre-training model and achieve the final training model after 300 epochs. All the parameter settings are consistent with the reference documents above.

III. Performance Evaluation

Performance evaluation is conducted in this section to demonstrate our DeepCoder. Meanwhile, we also provide the comparison with the well-known H.264/AVC [13] that is realized using the popular x264 [14]. We use the default x264 but with few modification for fair comparison with our DeepCoder. Details are listed in Table I.

TABLE I
x264 Encoder and DeepCoder Parameters

	x264	DeepCoder
Input Source	RGB 8 bits per sample	
GoP	IPPP	
References #	1	
Basic Unit	16×16	32×32
Prediction Unit	All	32×32
Transform Unit	All	N/A
ME Range	[-16, 16] (default)	[-10,10]
Quantization	scalar	
Interpolation	disabled	
Entropy Coding	CABAC	Huffman Codes

Six common test sequences that are used by video coding standardization group, are chosen to demonstrate the coding efficiency of our proposed DeepCoder and x264. As presented in Fig. 3, these videos cover typical content scenarios in practices, including richer texture scenes, such as BQSquare, BlowingBubbles, ParkScene and ParkScene; motion scenes (possibly with camera panning), such as RaceHorses; and stationary conferencing circumstances, e.g., KristenAndSara.

Fig. 4 shows the SSIM versus Bit Rate (kbps) obtained by both our DeepCoder and x264 based H.264/AVC. Bjontegaard Delta-Rate between two curves is calculated using [15]. Note that the distortion is measured by SSIM rather conventional PSNR. As shown in Table II, our DeepCoder has shown similar efficiency (with averaged BD-Rate loss less than 2%) compared with the well-known H.264/AVC.

IV. Conclusion

We present the DeepCoder - a neural network based video coding framework in this work. We stand upon the assumption that any signal can be presented by its prediction and corresponding residual, each of which is represented using designated convolutional neural networks. Operations are performed upon fixed 32×32 blocks. Scalar quantization and Huffman codes are directly applied . In comparison to the x264, our DeepCoder has shown very similar coding efficiency (but with little loss) with the common test sequences used by the video coding society, but it promises the great potential that deep neural network based video compression is an alternative for future video coding.

TABLE II
BD-Rate of DeepCoder vs. x264

Sequence	Resolution	FPS	BD-Rate Loss
BQSquare	416x240	60	1.01%
BlowingBubbles	416x240	50	2.12%
RaceHorses	832x480	30	1.68%
KristenAndSara	1280x720	60	1.91%
ParkScene	1920x1080	24	0.03%
Cactus	1920x1080	50	0.17%

Going forward, there are several potential avenues that we can explore to further improve our DeepCoder. For instance, recent prevalent Generative Adversarial Network (GAN) or Recurrent Neural Network (RNN) can be also investigated. Motion-compensation interpolation [16] has shown quite impressive performance improvement for conventional video coding. Neural network based interpolation is another interesting aspect to study.

Acknowledgment

This work was supported in part by the National Natural Science Foundation of China (NSFC) projects (Grant #s: 61371166, 61422107 and 61571215), in part by the National Science Foundation for Young Scholar of Jiangsu Province, China (Grant # BK20140610). Dr. Z. Ma is the corresponding author of this paper.

References

[1] G. J. Sullivan and T. Wiegand, "Video compression—from concepts to the H.264/AVC standard," *Proceedings of the IEEE*, vol. 93, no. 1, pp. 18–31, 2005.
[2] P. Schmid-Saugeon and A. Zakhor, "Dictionary design for matching pursuit and application to motion-compensated video coding," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 14, no. 6, pp. 880–886, 2004.
[3] J. Zepeda, C. Guillemot, and E. Kijak, "Image compression using sparse representations and the iteration-tuned and aligned dictionary," *IEEE J. Selected Topics in Signal Processing*, vol. 5, no. 5, pp. 1061–1073, 2011.
[4] Y. Xue and Y. Wang, "Video coding using a self-adaptive redundant dictionary consisting of spatial and temporal prediction candidates," in *Proc. IEEE Int. Conf. Multimedia and Expo (ICME)*, 2014.
[5] A. Atreya and D. O'Shea, "Novel lossy compression algorithms with stacked autoencoders," Stanford University CS229, Tech. Rep., 2009.
[6] 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," *CoRR*, vol. abs/1511.06085, 2015. [Online]. Available: http://arxiv.org/abs/1511.06085
[7] G. Toderici, D. Vincent, N. Johnston, S.-J. Hwang, D. Minnen, J. Shor, and M. Covell, "Full resolution image compression with recurrent neural networks," *CoRR*, vol. abs/1608.05148, 2016. [Online]. Available: http://arxiv.org/abs/1608.05148
[8] Z. Wang, A. C. Bovik, H. Sheikh, and E. P. Simoncelli, "Image quality assessment: from error visibility to structural similarity," *IEEE Trans. Image Processing*, vol. 13, no. 4, pp. 600–612, April 2004.
[9] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proc. of the IEEE CVPR*, 2015, pp. 1–9.
[10] K. Yu, C. Dong, C. C. Loy, and X. Tang, "Deep convolution networks for compression artifacts reduction," *arXiv:1608.02778*, Aug. 2016.
[11] M. D. Zeiler, "ADADELTA: An adaptive learning rate method," *arXiv:1212.5701v1*, Dec. 2012.
[12] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv:1412.6980v9*, Jan. 2017.
[13] T. Wiegand, G. J. Sullivan, G. Bjontegaard, and A. Luthra, "Overview of the H.264/AVC video coding standard," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 13, no. 7, pp. 560–576, 2003.
[14] "x264," http://www.videolan.org/developers/x264.html, 2016.
[15] G. Bjontegaard, "Calculation of average PSNR differences between R-D curves," in *Doc. VCEG-M33, ITU-T VCEG 13th Meeting*, 2001.
[16] B. Girod, "Motion-compensating prediction with fractional pel accuracy," *IEEE Trans. Commun.*, vol. 41, no. 4, pp. 604–612, Apr. 1993.