Advanced Learning-Based Coding Tools for ECM: Intra Prediction and In-Loop Filtering

Yanchen Zhao[1], Jiaye Fu[2], Zhaoyu Li[1], Qizhe Wang[1], Zhimeng Huang[1],
Jiaqi Zhang[1], Chuanmin Jia[3] and Siwei Ma[1]
[1]School of Computer Science, Peking University, China
[2]School of Electronic and Computer Engineering, Peking University, China
[3]Wangxuan Institute of Computer Technology, Peking University, China
{yczhao, jyfu, zylee, qzwang}@stu.pku.edu.cn, {zmhuang, jqzhang, cmjia, swma}@pku.edu.cn

***Abstract*—Neural Network (NN)-based video coding technologies have emerged as a promising alternative to traditional methods, demonstrating significant advantages amidst the rapid advancements in video coding technology. This paper presents a hybrid video coding method based on the Enhanced Compression Model (ECM) developed by the Joint Video Exploration Team (JVET). We integrate two NN-based coding tools into the framework. Specifically, the proposed NN-based Intra Prediction (NNIP) method effectively models the nonlinear relationship between neighboring contextual information and the block to be predicted. The NN-based In-Loop Filtering (NNILF) method adaptively filters the luminance and chrominance components across various quality levels. Experimental results show that the NNIP and NNILF methods achieve 0.56% and 4.14% BD-rate savings for YCbCr components under the All Intra (AI) configuration compared to ECM-14.0. Under the Random Access (RA) configuration, the proposed method can achieve a 2.41% BD-rate saving for YCbCr components.**

***Index Terms*—Video coding, intra prediction, in-loop filtering, learning-based**

I. Introduction

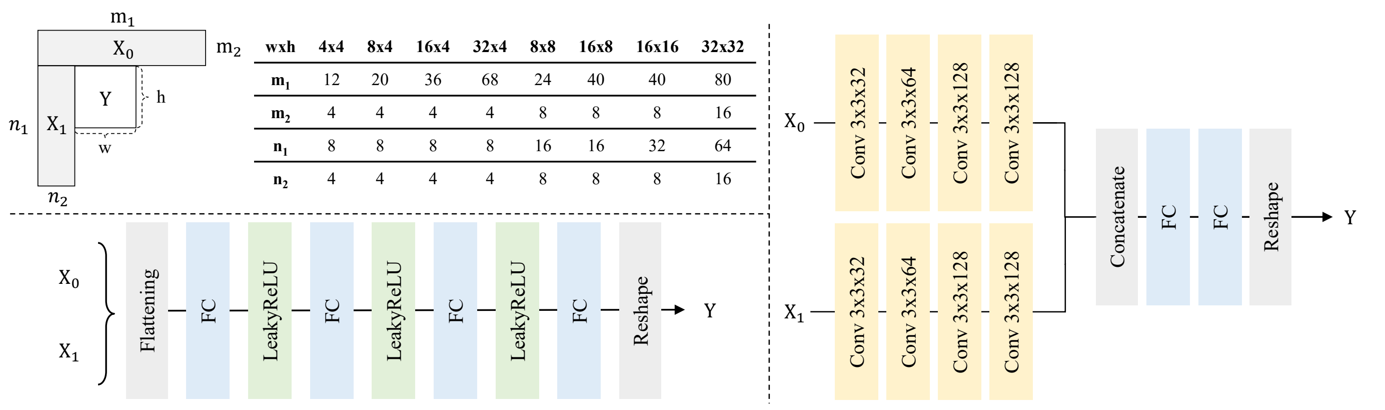
In recent years, video has gradually become the primary medium for information dissemination. With the development of High-Definition (HD) and Ultra-High-Definition (UHD) video, the volume of video data has been increasing exponentially, making the need for advanced video coding techniques more urgent than ever. Versatile Video Coding (VVC), introduced by the Joint Video Experts Team (JVET) in 2021, is the latest generation of video coding standards [1]. Compared to its predecessor, High Efficiency Video Coding (HEVC), VVC achieves a 50% Bjøntegaard Delta rate (BD-rate) saving with the comparative subjective video quality, while also supporting more types of media content and emerging applications [2]. In addition to VVC, JVET introduced a Neural Network Video Coding (NNVC) reference software in 2021 and Neural Compression Software (NCS) to explore how neural network-based tools can enhance coding performance. Furthermore, the Enhanced Compression Model (ECM) was introduced, incorporating various new tools to further improve video coding efficiency [3]. The initial version of ECM achieves a 12% BD-rate saving in the Random Access (RA) configuration compared with VTM-10.0, while the first version of NCS achieves a 9% BD-rate saving [4]. As of August 2024, the latest version, ECM-14.0, has achieved a coding performance improvement of over 24% compared to VTM-11.0 under the RA configuration, while NNVC-8.0 has achieved over 14% BD-rate saving under the RA configuration [5], [6].

This work was supported in part by National Natural Science Foundation of China 62025101, 62031013, 62371008, in part by New Cornerstone Science Foundation through the XPLORER PRIZE and in part by the High Performance Computing Platform of Peking University.

Beyond standards development [7], neural network-based coding tools are also being actively explored [8], [9]. In intra prediction, early neural network-based tools primarily used the Convolutional Neural Network (CNN) based method to predict coding blocks based on the reconstructed neighboring regions [10]. In recent years, with the rise of attention mechanisms, attention-based intra prediction has shown significant potential [11]. Temporal information plays a crucial role in video coding, and many neural network-based inter prediction methods have achieved high performance by leveraging spatio-temporal reference information across frames [12], [13]. In-loop filtering is also an important part of the video coding framework, and the introduction of neural network-based filtering methods has significantly improved coding efficiency [14]–[16]. Currently, the evolution of ECM has revealed diminishing returns in intra prediction tool improvements, primarily due to the inefficiency of traditional coding tools. Therefore, the incorporation of deep learning coding tools holds great promise for the development of next-generation video coding technologies.

Based on the advanced traditional coding reference platform ECM [17], this paper proposes a neural network-based video coding method, which incorporates two major neural network-based coding tools. This work represents a significant investigation into next-generation video coding from an artificial intelligence perspective. The main contributions of our work are summarized as follows:

- We propose a Neural Network-based Intra Prediction (NNIP) method into the intra prediction module of ECM, which provides an additional intra prediction mode. Compared with traditional methods, this method can generate non-linear and naturally transitional textures.
- We propose a Neural Network-based In-Loop Filtering (NNILF) method. This method is capable of filtering both luminance and chrominance components simultaneously,



wxh	4x4	8x4	16x4	32x4	8x8	16x8	16x16	32x32
m_1	12	20	36	68	24	40	40	80
m_2	4	4	4	4	8	8	8	16
n_1	8	8	8	8	16	16	32	64
n_2	4	4	4	4	8	8	8	16

Fig. 1. Illustration of the fully connected architecture and convolutional architecture. The fully connected network is used for predicting small blocks, and the convolutional network is used for predicting large blocks. For each prediction area Y of varying sizes, the required reference pixel area size also differs accordingly.

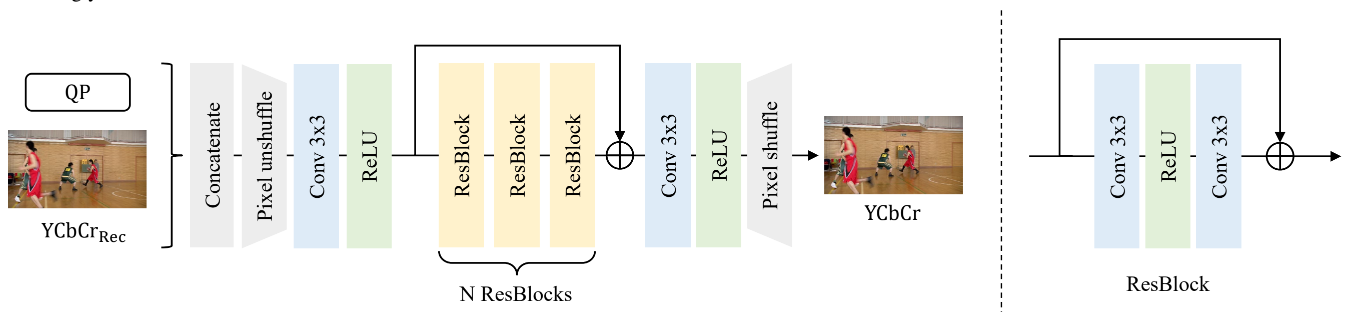


Fig. 2. Network architecture of in-loop filtering for luminance and chrominance components.

and can also achieve adaptive filtering by inputting the Quantization Parameter (QP) of the current frame.

II. Proposed Method

A. *Neural Network-based Intra Prediction*

Current intra prediction methods are mainly based on angular projection models. These models can generate linear textures but are limited when it comes to handling complex textures and producing natural transitions. The reference boundary pixels are prone to noise, which can result in prediction errors. Additionally, over 60 angular prediction modes require approximately 6 bits to represent, increasing bitstream overhead. On the other hand, neural networks have a large number of learnable parameters and a strong nonlinear fitting capability. This allows them to capture the characteristics of reference pixels and to generate transition textures that are more natural and complex. Furthermore, neural networks can generate diverse textures using a single prediction model, helping to reduce bit cost.

As shown in Fig. 1, we use two different network architectures to predict blocks of varying sizes. X_0 and X_1 represent the available reference pixel regions for the current block to be predicted, while Y represents the block to be predicted. The entire prediction process can be viewed as an end-to-end mapping from X_0 and X_1 to Y. Specifically, for smaller blocks, the number of pixels to be predicted is limited. In this case, using a fully connected network to map reference pixels to each predicted pixel achieves a good balance between complexity and model performance. For larger blocks, there are many pixels to predict. A fully connected network requires flattening the reference pixel region into a one-dimensional vector, which disrupts the local correlations between pixels. In contrast, convolutional networks are more adept at extracting local features and are thus better suited for predicting larger blocks. In this architecture, features are independently extracted by two branches and then concatenated to generate the predicted block. As shown in Fig. 1, we have trained a total of 8 sets of network models for 12 different block sizes. The values of m_1, m_2, n_1, and n_2, as well as the corresponding block sizes, are shown in the figure. For symmetric blocks, such as 4×8 and 8×4, we use the same fully connected network. For larger blocks, such as 16×16 and 32×32, we use the convolutional neural networks.

B. *Neural Network-based In-Loop Filtering*

To improve the inference efficiency of the in-loop filtering process in the traditional hybrid coding framework, we propose a network structure that simultaneously processes both luminance and chrominance components, as shown in Fig. 2. The model also takes Quantization Parameter (QP) as an additional input, allowing a single model to handle filtering tasks across multiple bitrate points. Specifically, YCbCr components are fed into the network at their original resolution. The chrominance components are first upsampled using nearest-neighbor interpolation to match the resolution of the luminance component. These are then concatenated with the luminance component and the QP, and passed through a pixel unshuffle layer to reduce the size of the feature maps for more efficient computation. A convolutional layer followed by a ReLU activation function performs initial feature extraction.

TABLE I
CODING PERFORMANCE OF NNIP
(BD-RATES AND RELATIVE RUNTIMES FOR ECM-14.0 CTC).

Class	All Intra					
	Y	Cb	Cr	YCbCr	EncT	DecT
A1	-0.58%	-0.48%	-0.51%	-0.56%	445%	778%
A2	-0.33%	-0.25%	-0.22%	-0.31%	446%	645%
B	-0.46%	-0.23%	-0.17%	-0.40%	487%	765%
C	-0.79%	-0.50%	-0.65%	-0.74%	440%	1019%
D	-0.84%	-0.64%	-0.76%	-0.81%	438%	1171%
E	-0.91%	-0.41%	-0.63%	-0.81%	494%	849%
F	-0.50%	-0.36%	-0.27%	-0.46%	424%	557%
TGM	-0.04%	0.00%	-0.04%	-0.03%	395%	224%
Overall	**-0.61%**	**-0.37%**	**-0.42%**	**-0.56%**	**463%**	**809%**

TABLE II
CODING PERFORMANCE OF NNILF
(BD-RATES AND RELATIVE RUNTIMES FOR ECM-14.0 CTC).

Class	All Intra					
	Y	Cb	Cr	YCbCr	EncT	DecT
A1	-1.91%	-6.37%	-7.06%	-3.10%	114%	17502%
A2	-2.20%	-8.31%	-8.11%	-3.70%	108%	12741%
B	-2.16%	-8.27%	-7.80%	-3.63%	109%	17890%
C	-2.95%	-9.60%	-10.21%	-4.69%	105%	13642%
D	-3.40%	-9.81%	-11.02%	-5.16%	105%	12512%
E	-4.20%	-9.55%	-11.23%	-5.75%	114%	20871%
F	-1.62%	-3.33%	-2.06%	-1.89%	105%	15672%
TGM	-0.66%	-0.01%	-0.03%	-0.50%	105%	14991%
Overall	**-2.64%**	**-8.47%**	**-8.83%**	**-4.14%**	**110%**	**16272%**

The extracted features are progressively enhanced through stacked residual blocks, followed by another convolutional layer and ReLU for feature reconstruction. Finally, a pixel shuffle layer restores the output to the original luminance resolution, while the chrominance components are downsampled using bilinear interpolation. This design allows the network to enhance both luminance and chrominance components in a single inference pass. When utilizing GPU resources for inference, this method effectively reduces the data transfer latency between system memory and GPU memory, compared to performing separate inferences for luminance and chrominance or relying on third-party upsampling functions. Additionally, compared to methods that split the luminance component into four parts and concatenate it with the chrominance components, our approach better preserves the spatial continuity of the luminance component, enabling the model to more effectively learn the correlations between the luminance and chroma components.

C. Implementation in ECM

We conduct experiments on the next-generation video codec Enhanced Compression Model (ECM). We have integrated the two proposed intelligent encoding tools into the latest reference software ECM-14.0 [18]. Specifically, NNIP, as an independent intra prediction coding tool, has a separate Coding Unit (CU) level flag. Currently, in order to ensure that each coding block has locally optimal transformation parameters, repetitive calculations are performed for the same prediction mode at the encoder side. However, the calculation of neural network can be extremely time-consuming. Therefore, the proposed method introduces an additional cache buffer at the encoder side to store the prediction results of NNIP, thereby avoiding the redundancy in time complexity caused by repetitive neural network calculations.

We integrate the neural network-based in-loop filtering tool after the traditional deblocking filter to efficiently combine it with conventional filtering tools. During in-loop filtering of the reconstructed frame, the proposed approach applies block-by-block filtering to each Coding Tree Unit (CTU) of the current reconstructed frame, then calculates the Mean Square Error (MSE) between the CTU block before and after filtering and the original block. We design additional syntax elements at both the frame level and CTU level. The frame level is represented by 3 bits, indicating the activation status of the filtering tool for the Y, Cb, and Cr components. When frame-level filtering is enabled, additional flags are transmitted to further indicate the activation status of the filtering tool for each CTU block, enabling more precise filtering.

III. EXPERIMENTAL RESULTS

A. Training Process

We use DIV2K [19], Flickr2K [19], and BVI-DVC [20] as the training datasets. We convert the original PNG format images and MP4 format videos into 10-bit YUV420 format videos by FFmpeg. For the training of the intra prediction models, we use ECM-14.0 to compress the DIV2K and Flickr2K datasets at four different Quantization Parameters (QPs) (22, 27, 32, 37) under All Intra (AI) configuration. Other parameters are the same as Joint Video Exploration Team (JVET) Common Test Condition (CTC) [21]. Then we export the reference pixels of eight different block sizes along with the original values of these blocks to form the training set. The validation set consists of the last 10% images from these two datasets. Due to the large amount of data exported from blocks of different sizes, we set the batch size to 1000 and the learning rate to 3e-5 when training the model. Each model is trained for 90 epochs, and the learning rate is reduced to one-third of its original value every 30 epochs. Adam [22] is used as the optimizer.

For the training of the filtering model, we conduct the training in two stages. The first stage involves training the model under the AI configuration, using the DIV2K and Flickr2K datasets. We switch the Luma Mapping with Chroma Scaling (LMCS), Deblocking Filter (DBF), Sample Adaptive Offset (SAO), Adaptive Loop Filter (ALF), and Bilateral Filter (BIF) off during the compression process. We compress these datasets at four different QPs (22, 27, 32, 37) using ECM-14.0 under AI configuration. We also export the QP for each frame as an additional input to the model. The input size of the model is 144×144, with a batch size set to 32 and an initial learning rate of 1e-4. The model is trained for a total of 60 epochs, and the learning rate is reduced to one-third of its original value every 20 epochs. In the second stage, we compress the training set under the RA configuration using the BVI-DVC dataset, which leads to some performance loss in the model. We encode the first 64 frames of each video under RA configuration with a Group of Pictures (GOP) size of 32. We extract all inter-frames from the dataset to form

TABLE III
CODING PERFORMANCE OF NNIP AND NNILF
(BD-RATES AND RELATIVE RUNTIMES FOR ECM-14.0 CTC).

Class	Sequences	**All Intra**						**Random Access**					
		Y	Cb	Cr	YCbCr	EncT	DecT	Y	Cb	Cr	YCbCr	EncT	DecT
A1	Tango	-2.88%	-11.96%	-13.62%	-5.36%	443%	20708%	-1.84%	-9.69%	-7.22%	-3.49%	125%	21786%
	FoodMarket	-3.06%	-7.95%	-8.32%	-4.33%	442%	19234%	-1.23%	-5.39%	-4.20%	-2.12%	139%	26143%
	CampFire	-1.31%	-0.23%	-0.54%	-1.08%	425%	16453%	-0.65%	-0.96%	-0.09%	-0.62%	105%	24855%
A2	CatRobot	-3.35%	-11.13%	-13.26%	-5.56%	443%	14141%	-3.02%	-8.24%	-7.96%	-4.29%	116%	22517%
	DaylightRoad	-2.10%	-14.11%	-11.37%	-4.76%	512%	13743%	-	-	-	-	-	-
	ParkRunning	-2.00%	-0.37%	-0.43%	-1.60%	367%	14080%	-2.99%	-1.26%	-2.19%	-2.67%	110%	15109%
B	MarketPlace	-2.12%	-9.34%	-7.12%	-3.65%	500%	24761%	-2.14%	-13.89%	-10.40%	-4.64%	113%	16421%
	RitualDance	-3.67%	-8.54%	-10.82%	-5.17%	493%	25044%	-2.78%	-6.57%	-7.02%	-3.78%	104%	20419%
	Cactus	-2.68%	-1.28%	-0.87%	-2.28%	481%	17380%	-2.02%	-1.90%	0.06%	-1.74%	114%	21413%
	BasketballDrive	-2.60%	-12.70%	-10.52%	-4.85%	498%	19116%	-1.76%	-6.36%	-5.35%	-2.79%	107%	18735%
	BQTerrace	-1.75%	-10.39%	-10.40%	-3.91%	511%	12564%	-2.24%	-9.67%	-10.59%	-4.21%	120%	21944%
C	BasketballDrill	-4.72%	-13.08%	-12.47%	-6.73%	467%	14739%	-1.45%	-3.86%	-2.30%	-1.85%	110%	19118%
	BQMall	-4.42%	-10.93%	-12.43%	-6.23%	465%	16680%	-2.13%	-8.44%	-7.50%	-3.59%	102%	15837%
	PartyScene	-3.16%	-8.57%	-7.96%	-4.43%	436%	9619%	-1.86%	-6.22%	-4.90%	-2.78%	102%	11735%
	RaceHorses	-2.29%	-7.78%	-10.67%	-4.02%	443%	18852%	-1.06%	-7.87%	-10.03%	-3.03%	104%	10335%
D	BasketballPass	-4.63%	-10.57%	-10.64%	-6.12%	444%	15920%	-2.41%	-7.50%	-7.94%	-3.74%	110%	15275%
	BQSquare	-4.66%	-7.77%	-13.09%	-6.10%	451%	10660%	-4.42%	-6.70%	-9.01%	-5.28%	108%	15070%
	BlowingBubbles	-3.53%	-9.89%	-8.72%	-4.97%	415%	11944%	-1.60%	-7.53%	-6.27%	-2.93%	107%	10746%
	RaceHorses	-3.71%	-13.33%	-14.73%	-6.29%	422%	16120%	-1.75%	-10.58%	-10.89%	-4.00%	105%	13289%
E	FourPeople	-5.36%	-10.10%	-10.33%	-6.58%	518%	19580%	-	-	-	-	-	-
	Johnny	-4.80%	-9.94%	-14.75%	-6.68%	507%	22616%	-	-	-	-	-	-
	KristenAndSara	-4.62%	-9.87%	-10.86%	-6.06%	496%	23273%	-	-	-	-	-	-
F	BasketballDrillText	-4.38%	-5.64%	-0.08%	-4.00%	417%	13096%	-1.43%	-3.71%	0.61%	-1.46%	139%	31837%
	ArenaOfValor	-3.09%	-8.80%	-9.27%	-4.58%	421%	15476%	-1.25%	-4.66%	-4.61%	-2.09%	132%	24980%
	SlideEditing	-0.14%	-0.18%	0.21%	-0.10%	459%	16257%	-0.07%	-0.52%	-0.05%	-0.13%	150%	87807%
	SlideShow	-0.60%	0.01%	-0.42%	-0.50%	456%	27330%	-0.08%	-1.60%	-1.55%	-0.45%	145%	70996%
TGM	FlyingGraphic	-1.48%	-0.14%	-0.19%	-1.14%	386%	13385%	-0.53%	-0.39%	-0.82%	-0.55%	104%	16742%
	Desktop	0.10%	0.15%	0.04%	0.10%	404%	20125%	-0.16%	-0.13%	-0.29%	-0.17%	152%	63910%
	Console	-1.18%	-0.10%	-0.09%	-0.91%	342%	18556%	-0.09%	0.07%	0.10%	-0.05%	124%	44914%
	ChineseEditing	-0.29%	0.04%	-0.05%	-0.22%	448%	12650%	-0.27%	0.12%	0.15%	-0.17%	143%	85847%
	Overall	**-3.16%**	**-8.79%**	**-9.26%**	**-4.63%**	**468%**	**17398%**	**-1.59%**	**-5.13%**	**-4.63%**	**-2.41%**	**119%**	**28763%**

the training set. Additionally, unlike the first stage, the initial learning rate in the second stage is set to 1e-5, while all other parameters remain the same.

B. Coding Gain Analysis

The proposed method is integrated into ECM-14.0. Simulations are conducted following the JVET CTC and evaluation procedures for enhanced compression tool testing. We test four QPs (22, 27, 32, 37), and the comparison of coding efficiency is measured by BD-rate. As shown in Tab. I, the NNIP method achieves average BD-rate savings of 0.61%, 0.37%, and 0.42% under the AI configuration, compared to ECM-14.0. Tab. II shows that the NNILF method provides average BD-rate savings of 2.64%, 8.47%, and 8.83% under AI configuration. Tab. III summarizes the overall performance comparison, where the combined methods achieve 3.16%, 8.79%, and 9.26% BD-rate savings on average under AI configuration, and 1.59%, 5.13%, and 4.63% under RA configuration. The YCbCr BD-Rate is calculated by averaging the BD-Rate of Y, Cb, and Cr components with weights 6:1:1. It is worth noting that the models are not trained with screen content-related data, resulting in suboptimal performance on screen content, such as Class F and Class TGM.

C. Coding Complexity Analysis

For the NNIP method, the number of models' parameters ranges from 1.5M to 4M. The computational complexity of the fully connected networks varies from 5 kilo Multiply-Accumulate operations (kMACs) per pixel to 20 kMACs per pixel. The computational complexity of the convolutional neural networks is around 200 kMACs per pixel. When tested under the AI configuration, we find that the computational complexity mainly comes from the two convolutional networks applied to large blocks of 16×16 and 32×32 sizes. Therefore, in the RA configuration, we disable these two convolutional networks to minimize the encoding and decoding complexity under RA. For the NNILF method, the model is primarily composed of 20 stacked residual blocks, with each convolutional layer having 32 channels. The total number of parameters in the network is 376.9K, and the computational complexity is 93.9 kMACs per pixel. None of the proposed neural network models are quantized, and all are inferred using the CPU, which leads to higher encoding and decoding complexity.

IV. CONCLUSION

In this paper, we integrate Neural Network-based Intra Prediction (NNIP) and Neural Network-based In-loop Filtering (NNILF) into ECM-14.0. Experimental results show that these tools can achieve 4.63% BD-Rate savings for YCbCr components under the All Intra (AI) configuration and 2.41% under the Random Access (RA) configuration. Neural network-based coding tools possess substantial potential for future development. In the future, these tools will effectively support the development of next-generation video coding standards.

References

[1] B. Bross, Y.-K. Wang, Y. Ye, S. Liu, J. Chen, G. J. Sullivan, and J.-R. Ohm, “Overview of the Versatile Video Coding (VVC) standard and its applications,” *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 31, no. 10, pp. 3736–3764, 2021.

[2] G. J. Sullivan, J.-R. Ohm, W.-J. Han, and T. Wiegand, “Overview of the High Efficiency Video Coding (HEVC) Standard,” *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 22, no. 12, pp. 1649–1668, 2012.

[3] Y.-J. Chang, C.-C. Chen, J. Chen, J. Dong, H. E. Egilmez, N. Hu, H. Huang, M. Karczewicz, J. Li, B. Ray, K. Reuze, V. Serigin, N. Shlyakhov, L. P. Van, H. Wang, Y. Zhang, and Z. Zhang, “Compression Efficiency Methods beyond VVC,” *document JVET-U0100*, 2021.

[4] Y. Li, J. Li, C. Lin, K. Zhang, L. Zhang, F. Galpin, T. Dumas, H. Wang, M. Coban, J. Ström *et al.*, “Designs and Implementations in Neural Network-based Video Coding,” *arXiv preprint arXiv:2309.05846*, 2023.

[5] F. Galpin, R. Chang, and Y. e. a. Li, “JVET AHG report: NNVC software development AhG14,” *document JVET-AJ0014*, 2024.

[6] V. Seregin, J. Chen, R. Chernyak, F. Le Leannec, and K. Zhang, “JVET AHG report: ECM software development (AHG6),” *document JVET-AH0006*, 2024.

[7] S. Ma, L. Zhang, S. Wang, C. Jia, S. Wang, T. Huang, F. Wu, and W. Gao, “Evolution of AVS video coding standards: twenty years of innovation and development,” *Science China Information Sciences*, vol. 65, no. 9, p. 192101, 2022.

[8] S. Ma, X. Zhang, C. Jia, Z. Zhao, S. Wang, and S. Wang, “Image and video compression with neural networks: A review,” *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 30, no. 6, pp. 1683–1698, 2019.

[9] S. Ma, J. Gao, R. Wang, J. Chang, Q. Mao, Z. Huang, and C. Jia, “Overview of intelligent video coding: from model-based to learning-based approaches,” *Visual Intelligence*, vol. 1, no. 1, p. 15, 2023.

[10] T. Dumas, A. Roumy, and C. Guillemot, “Context-adaptive neural network-based prediction for image compression,” *IEEE Transactions on Image Processing*, vol. 29, pp. 679–693, 2019.

[11] M. G. Blanch, S. Blasi, A. Smeaton, N. E. O’Connor, and M. Mrak, “Chroma Intra Prediction with Attention-based CNN Architectures,” in *2020 IEEE International Conference on Image Processing (ICIP)*. IEEE, 2020, pp. 783–787.

[12] P. Merkle, M. Winken, J. Pfaff, H. Schwarz, D. Marpe, and T. Wiegand, “Spatio-Temporal Convolutional Neural Network for Enhanced Inter Prediction in Video Coding,” *IEEE Transactions on Image Processing*, vol. 33, pp. 4738–4752, 2024.

[13] L. Zhao, S. Wang, X. Zhang, S. Wang, S. Ma, and W. Gao, “Enhanced Motion-Compensated Video Coding With Deep Virtual Reference Frame Generation,” *IEEE Transactions on Image Processing*, vol. 28, no. 10, pp. 4832–4844, 2019.

[14] Z. Huang, J. Sun, X. Guo, and M. Shang, “Adaptive Deep Reinforcement Learning-Based In-Loop Filter for VVC,” *IEEE Transactions on Image Processing*, vol. 30, pp. 5439–5451, 2021.

[15] C. Jia, S. Wang, X. Zhang, S. Wang, J. Liu, S. Pu, and S. Ma, “Content-Aware Convolutional Neural Network for In-Loop Filtering in High Efficiency Video Coding,” *IEEE Transactions on Image Processing*, vol. 28, no. 7, pp. 3343–3356, 2019.

[16] Y. Li, L. Zhang, and K. Zhang, “IDAM: Iteratively trained deep in-loop filter with adaptive model selection,” *ACM Transactions on Multimedia Computing Communications and Applications*, vol. 19, no. 1, 2023.

[17] “ECM-14.0 software repository,” https://vcgit.hhi.fraunhofer.de/ecm/ECM/-/tree/ECM-14.0.

[18] M. Coban, R.-L. Liao, and K. Naser, “Algorithm description of Enhanced Compression Model 14 (ECM 14),” in *JVET-AI2025, Sapporo*, 2024.

[19] E. Agustsson and R. Timofte, “Ntire 2017 challenge on single image super-resolution: dataset and study,” in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition Workshops*, 2017, pp. 126–135.

[20] D. Ma, F. Zhang, and D. Bull, “BVI-DVC: a training database for deep video compression,” *IEEE Transactions on Multimedia*, 2021.

[21] M. Karczewicz and Y. Ye, “Common Test Conditions and Evaluation Procedures for Enhanced Compression Tool Testing,” *document JVET-AI2017*, 2024.

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