On Content-Aware Post-Processing: Adapting Statistically Learned Models to Dynamic Content

YICHI ZHANG, GONGCHUN DING, and DANDAN DING, Hangzhou Normal University, China
ZHAN MA, Nanjing University, China
ZHU LI, University of Missouri–Kansas City, USA

Learning-based post-processing methods generally produce neural models that are statistically optimal on their training datasets. These models, however, neglect intrinsic variations of local video content and may fail to process unseen content. To address this issue, this article proposes a content-aware approach for the post-processing of compressed videos. We develop a backbone network, called *BackboneFormer*, where a Fast Transformer using Separable Self-Attention, Spatial Attention, and Channel Attention is devised to support underlying feature embedding and aggregation. Furthermore, we introduce Meta-learning to strengthen BackboneFormer for better performance. Specifically, we propose Meta Post-Processing (Meta-PP) which leverages the Meta-learning framework to drive BackboneFormer to capture and analyze input video variations for spontaneous updating. Since the original frame is unavailable to the decoder, we devise a Compression Degradation Estimation model where a low-complexity neural model and classic operators are used collaboratively to estimate the compression distortion. The estimated distortion is then utilized to guide the BackboneFormer model for dynamic updating of weighting parameters. Experimental results demonstrate that the proposed BackboneFormer itself gains about 3.61% Bjøntegaard delta bit-rate reduction over Versatile Video Coding in the post-processing task and "BackboneFormer + Meta-PP" attains 4.32%, costing only 50K and 61K parameters, respectively. The computational complexity of MACs is 49k/pixel and 50k/pixel, which represents only about 16% of state-of-the-art methods having similar coding gains.

CCS Concepts: • **Computing methodologies** → **Image compression**; *Reconstruction;*

Additional Key Words and Phrases: VVC, in-loop filtering, post-processing, transformer, Meta-learning

ACM Reference format:
Yichi Zhang, Gongchun Ding, Dandan Ding, Zhan Ma, and Zhu Li. 2023. On Content-Aware Post-Processing: Adapting Statistically Learned Models to Dynamic Content. *ACM Trans. Multimedia Comput. Commun. Appl.* 20, 1, Article 28 (September 2023), 23 pages.
https://doi.org/10.1145/3612925

This research was supported by the National Natural Science Foundation of China (62171174) and National Undergraduate Training Program for Innovation and Entrepreneurship.
Authors' addresses: Y. Zhang, G. Ding, and D. Ding (corresponding author), Hangzhou Normal University, Hangzhou, China; emails: 1ch.zhang233@gmail.com, 2021112011023@stu.hznu.edu.cn, DandanDing@hznu.edu.cn; Z. Ma, Nanjing University, Nanjing, China; email: mazhan@nju.edu.cn; Z. Li, University of Missouri–Kansas City, Kansas City; email: zhu.li@ieee.org.

1551-6857/2023/09-ART28 $15.00
https://doi.org/10.1145/3612925

1 INTRODUCTION

Compression artifacts severely affect the objective and subjective quality of reconstructed videos. To attenuate these artifacts, in-loop filtering and post-processing techniques are developed. The in-loop filters are applied in the encoding and decoding loop as standard tools [44], typically including the Deblocking Filter (DBF) [40] for blocking artifact alleviation, the Sample Adaptive Offset (SAO) [13] for ringing artifact removal, and the Adaptive Loop Filter (ALF) [26, 45] for other artifact attenuation. The post-processing filters, instead, are optionally applied at the decoder side to enhance the quality of decoded frames.

1.1 Background

In the past few years, **Convolutional Neural Network (CNN)**-based filters have exhibited significant performance advantages over traditional in-loop and post-processing filters in video coding. A variety of network structures [6, 7, 16, 21, 22, 24, 25, 29, 31, 34, 41, 47, 54] have been devised to exploit the spatial/temporal correlations within a frame or across frames for effective representation. Most methods pre-train their neural models offline for statistical, global optimum on a massive dataset and then apply these models directly for inference. Even though video content may dynamically change and exhibit different characteristics, these models will not adjust themselves during the filtering process. For a frame with unique variations that are unseen in the training dataset, the best solution for these methods is locally optimal [21, 24, 30]. Consequently, pre-trained models may perform unsatisfactorily in such scenarios, as demonstrated in state-of-the-art **Super-Resolution (SR)** [42, 43], style-transferring [4], and inpainting [50] studies. Clearly, both the external training dataset and the internal content variations should be taken into account in a neural model for distortion adaptive compensation (i.e., content-aware models are desired). The essential problem thus becomes how to make neural models adaptive to local video variations.

When the original, uncompressed videos are known (e.g., at the encoder side), we can encapsulate some prior knowledge as side information and send it to the decoder side to guide the neural model for filtering [8, 28]. In our previous work [8, 28], we used a CNN model to project the degraded frame into an M-dimensional space, generating M sub-signals; then we linearly combined these sub-signals using weighting coefficients to adaptively compensate for the compression distortion. The weighting coefficients, which are explicitly transmitted in the bitstream, were estimated by minimum **Mean Squared Error (MSE)** optimization, guiding the neural model to fit local content variations. Some methods directly apply the enhancement strategies following the CNN model for video-content-specific fine-tuning—for example, pixel values can be further scaled [54] or filtered by a Wiener filter [32]. However, these tools often work as separate parts after CNNs and cannot reach the collaborative effect together with CNN models.

On the contrary, at the decoder side, the original videos are unavailable, and no side information about the current frame is delivered in advance. As a result, the post-processing becomes an ill-posed problem, as the decoder cannot accurately know the variations of the current frame. To address this issue, most existing methods enhance compressed frames uniformly using a pre-trained model without any content-aware adaption [6, 16, 25, 29, 31, 34, 41, 47, 54]. However, Xing et al. [57] found that not all frames need to go through the whole model. Given a pre-trained model, they proposed RBQE (Resource-efficient Blind Quality Enhancement), which decided to terminate or continue the filtering task according to the assessed quality of enhanced frames. However, RBQE does not explicitly utilize the video content but only the non-reference quality assessment algorithm for a rough decision, which thus limits its performance. In addition, its high performance relies on large-scale models, leading to extremely high computational complexity (ranging from 18 to 28 GMACs).

For the post-processing of compressed videos, it remains a great challenge to adapt statistically learned models to dynamic content for high filtering performance. To address this issue, this article proposes a Meta-learning-based content-aware approach.

1.2 Approach and Contribution

Unlike in the work of Xing et al. [57], in which a network structure is selected for each frame during testing, the proposed method aims to dynamically change the parameters of a neural model in testing according to input content variations. The key challenges are (1) how to capture video variations since the distortion to the original video is unknown and (2) how to update the neural model using such variations.

To address these challenges, the proposed approach develops three essential components: (1) a backbone network called *BackboneFormer*, which provides underlying support of content-adaptive post-processing; (2) a **Compression Degradation Estimation (CDE)** model to approximate compression artifacts of the current input frame; and (3) a **Meta Post-Processing (Meta-PP)** framework that integrates BackboneFormer and CDE for collaborative training and testing and enables parameter updating of BackboneFormer according to the current input on the fly.

More specifically, BackboneFormer utilizes multiple attention mechanisms for intelligent feature extraction and aggregation. It is equipped with Self-Attention, Spatial Attention, and Channel Attention to best exploit pixel correlations for global feature embedding. This BackboneFormer provides a basic network structure to build models of different sizes (e.g., models with 50K parameters and 1,500K parameters in this work) for post-processing. To tackle the first challenge, we devise a CDE model incorporating both classic operators and the CNN-based model for accurate CDE. To reduce the computational complexity and speedup the estimation process, the CNN-based model adopts a U-Net style with only 11K parameters and 1K MACs/pixel complexity. To tackle the second challenge, we resort to the Meta-learning framework, which involves both BackboneFormer and CDE for collaborative training: in the outer loop training, BackboneFormer and CDE are separately trained with respective loss functions, and in the inner loop training, the BackboneFormer is further fine-tuned using the fixed CDE model. Such alternating execution of outer and inner training fully utilizes training information to polish model parameters. In addition, in the testing stage, we fine-tune BackboneFormer again using input compressed samples. In this way, the updated BackboneFormer immediately grasps variations of input local content and adapts itself for better processing.

In experiments, we exemplify a small-scale BackboneFormer model having only 50K parameters. Results show that this BackboneFormer model costs about 50K MACs/pixel, representing only 16% compared to the complexity of state-of-the-art methods having the same coding performance. A larger-scale BackboneFormer (BackboneFormer Ext.) can be constructed by simply stacking more attention blocks. Their performance compared with state-of-the-art works is illustrated in Figure 1.

Our main contributions are summarized as follows:

- We build a content-aware backbone network, called *BackboneFormer*, for the underlying support of compressed video post-processing. BackboneFormer is mainly comprised of stacked Fast Transformers, where Separable Self-Attention is embedded along with Spatial Attention and Channel Attention for global feature aggregation and embedding.
- We propose the Meta-PP framework, which leverages Meta-learning to drive BackboneFormer to adapt to both external and internal content variations. Since the original frames are unavailable, a learning-based CDE is devised to estimate the compression distortion. CDE is then used to guide BackboneFormer for parameter updating according to local content dynamics.

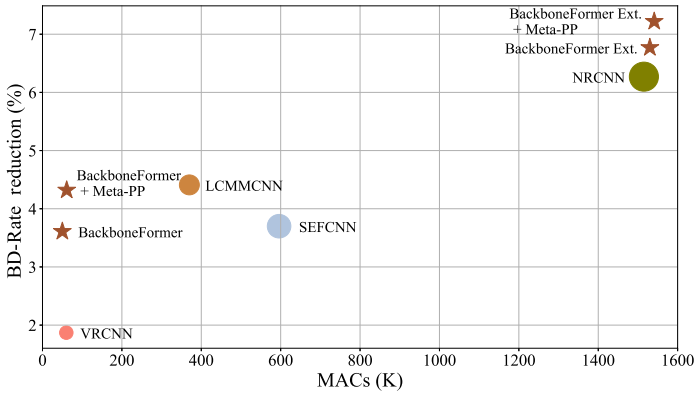


Fig. 1. Compression performance and computation complexity comparison with state-of-the-art works on the VVC platform. All compared methods, including VRCNN [6], SEFCNN [7], LCMMCNN [49], and NRCNN [31], are trained and tested using the same settings for a fair comparison.

- We conduct extensive experiments to demonstrate the effectiveness of the proposed approach. Results show that BackboneFormer itself attains a 3.61% **Bjøntegaard Delta Bit-Rate (BD-Rate)** reduction over the **Versatile Video Coding (VVC)** anchor with only 50K parameters. Furthermore, using "BackboneFormer + Meta-PP" improves the coding gains to 4.32%. The corresponding computational complexity of MACs is 50k/pixel, representing only about 16% of state-of-the-art methods having the same coding gains.

2 RELATED WORK

2.1 Post-Processing for Compressed Videos

Since the performance of learning-based post-processing has significantly surpassed that of traditional filters, we mainly focus on learning-based approaches.

Early learning-based restoration methods were mainly devised to remove image coding artifacts [9, 14]. Later, CNNs were widely used for either video in-loop filtering or post-processing to enhance the quality of compressed videos [6, 7, 16, 21, 22, 24, 25, 29, 31, 34, 41, 47, 54]. Taking the latest VVC standard (under All Intra configuration) as an example, Huang et al. [21] proposed an adaptive reinforcement learning filter. This filter modeled the filtering process as a decision process and used deep reinforcement learning to select an appropriate network. As a result, the adaptive reinforcement learning filter achieved on average 2.17% BD-Rate gains over the VTM-11.0 with a model size of 8.4 MB. Ma et al. [34] proposed a large-scale model called *MFRNet*, which was composed of four multi-level feature review residual dense blocks (MFRBs) and attained a 5.1% BD-Rate reduction over the VTM-7.0. Huang et al. [22] presented a variable convolutional network based VVC in-loop filtering model that achieved 3.63% BD-Rate reduction in the VTM-6.0 platform with 929.41K parameters. Li et al. [30] proposed a CNN model with a size of 5.6 MB, utilizing an iterative training scheme and a filter selection mechanism, which resulted in 7.91% BD-Rate reduction over the VTM-11.0. Furthermore, the Exploration Experiment in Joint Video Experts Team (JVET) [29] reported a CNN-based filter that gained 7.26% BD-Rate reduction against the VVC. It used a model size of 1.56M and a high MACs cost of 539k/pixel; moreover, its encoding time was 146% and decoding time was 24,974% of the VVC anchor.

As can be seen, the excellent performance of most previous CNN-based in-loop filtering or post-processing approaches is achieved at the expense of large-scale models with high computational complexity, requiring both large memory storage and an expensive GPU for running. Moreover,

all previous models are pre-trained offline and applied uniformly for filtering compressed frames, regardless of video content variations, which ignores the unique characteristics of each image and potentially fails to process unseen content in their training dataset.

2.2 Vision Transformers

The Transformer [46], which was first proposed for sequence processing, has been the obvious choice in the field of natural language processing. Its essential idea is to capture long-range dependencies in embedded data through global self-attention [17]. Due to its impressive performance, many Transformer variations (e.g., Swin Transformer [33], which employed a window-based attention mechanism, and Pyramid Vision Transformer (PVT) [51], which used a progressive shrinking pyramid and a spatial-reduction attention layer) have been devised to process high-level vision tasks. Recently, Transformers are also introduced to handle low-level vision tasks, such as texture transfer [59], cross-modal retrieval [37], image restoration [3], and enhancement [53]. More details can be found in the work of Han et al. [18].

Compared to conventional CNNs, Transformers typically have fewer parameters but suffer from increased latency and computational complexity due to the expensive procedures (e.g., batch matrix multiplication) for computing self-attention. The main efficiency bottleneck lies in the core component—**Multi-Head Self-Attention (MHSA)**, whose time complexity is closely related to the number of patches (tokens) (i.e., $O(k^2)$ for k patches).

2.3 Meta-Learning and Applications

In typical image/video filtering methods, the neural network is used to learn the non-linear mapping from samples to labels given a training dataset. Once the network is trained, it approximates a function that achieves the statistical optimum on this training dataset. However, this function may not be the best fit for every individual input image (or image region), especially when the input data have unique content variations and characteristics. To resolve this limitation, not only the external dataset (i.e., the training dataset) but also the image (or image region) itself (i.e., internal information) should be taken into account to improve the performance of the neural model. Then, the key challenge is how to jointly use external and internal information for instant model updating.

Numerous methods have been proposed to address this issue. Mosseri et al. [38] noted that different image patches have inherently different internal or external denoising preferences; they estimated the PatchSNR value for each image patch and decided to use either external or internal denoising based on the result. Bahat et al. [1] further suggested a blur-field estimation method to reblur the blurry images, which generated internal information to facilitate the deblurring algorithm. However, the estimation model is extremely sophisticated to solve.

The recently emerged Meta-learning framework provides a new way to integrate the external dataset and the internal image data. In Meta-learning, models are trained over a variety of learning tasks and optimized for the best performance on a distribution of tasks. As such, a model can quickly and efficiently adapt to new tasks that have never been encountered during training time. The prevalent optimization-based Meta-learning [11, 12, 15] is modeled as a two-level optimization problem: in the outer level, the cross-task performance is optimized; in the inner level, the base learner uses optimization strategies (e.g., gradient descent) for task-specific updates.

Using Meta-learning, many SR approaches have been proposed [20, 35, 42]. Hu et al. [20] developed Meta-SR. At arbitrary scale factors, they utilized the Meta-upscale module to predict the weights of the magnification filter from the input scaling factors and then used these weights to generate high-resolution images. Park et al. [42] also trained a network for SR through Meta-learning. In the testing phase, given a low-resolution image, the parameters of this Meta-learning network can be quickly fine-tuned to adapt to the new input in just a few iterations.

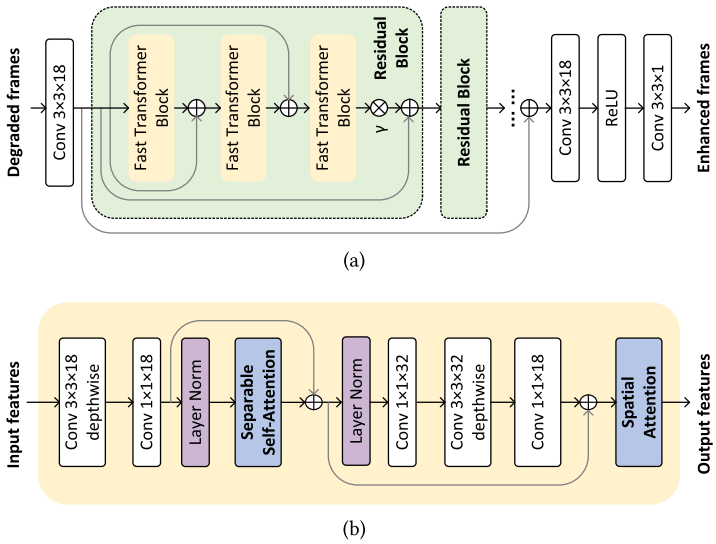


Fig. 2. Architecture of the proposed BackboneFormer. (a) Structure of BackboneFormer, which consists of cascading Residual Blocks. Each Residual Block contains three Fast Transformers. (b) Structure of Fast Transformer that embeds both Separable Self-Attention and Spatial Attention modules for feature aggregation. At the end of each Residual Block, a simplified Channel Attention is appended to further exploit channel correlations. As such, attention mechanisms are applied to the pixel level, local region level, and channel level for content-aware feature embedding as much as possible.

Previous works in image/video SR demonstrate the advantage of Meta-learning in addressing the limitations of conventional learning-based approaches that require diverse and sufficient raw data as well as complex manual tuning efforts. For the quality enhancement problem of compressed videos, since the original, uncompressed sources are unknown to the decoder, accurately tracking content variations becomes a challenge. In this regard, we develop the Meta-learning-based post-processing framework where our proposed backbone network is first trained with a large-scale dataset for external training and then fine-tuned using the estimated internal distortion information for performance optimization.

3 TRANSFORMER-BASED BACKBONEFORMER

This section details the proposed Transformer-based backbone network (called *BackboneFormer*). On top of this BackboneFormer, we implement the post-processing task in Section 4.

3.1 Framework of BackboneFormer

As shown in Figure 2(a), the proposed BackboneFormer sequentially goes through a plain convolutional layer of size $3 \times 3 \times 18$ for coarse feature extraction, a set of Residual Blocks for feature aggregation and embedding, and two plain convolutional layers again for reconstruction. Each Residual Block contains three Fast Transformers, as detailed in Figure 2(b).

3.2 Fast Transformer

Instead of collecting features with a fixed-size window in a frame like Swin Transformer [33], we devise Fast Transformer for global information aggregation on top of the whole frame, so as to perceive and capture valuable content characteristics as much as possible.

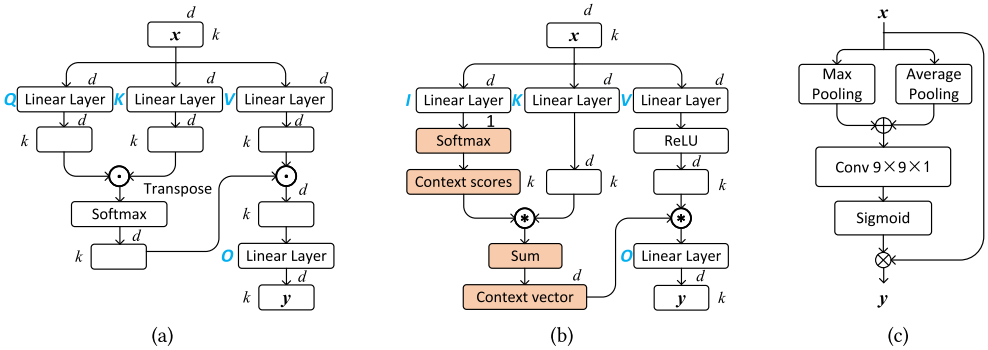


Fig. 3. Different attention strategies. (a) Structure of MHSA [10]. (b) Structure of Separable Self-Attention used in this work. (c) Structure of Spatial Attention.

The structure of our proposed Fast Transformer is illustrated in Figure 2(b). It consists of three core components: *Separable Self-Attention*, *Spatial Attention*, and *Channel Attention*. Specifically, input features first go through a $3 \times 3 \times 18$ depthwise convolution layer to keep each channel separate and a plain convolution layer with a kernel size of $1 \times 1 \times 18$ to mix features. The use of depthwise convolution also reduces computational complexity. After layer normalization, the latent features are sent into the Separable Self-Attention module, followed by a $1 \times 1 \times 32$ plain convolution layer, a $3 \times 3 \times 32$ depthwise convolution layer, and a $1 \times 1 \times 18$ plain convolution layer. At the end of each Fast Transformer, a Spatial Attention module is applied for further feature selection. After three Fast Transformers, the Channel Attention module is applied. As such, we deploy the attention mechanisms from different perspectives, enabling effective feature extraction and embedding.

Separable Self-Attention. As shown in Figure 3(a), the widely used MHSA employs dot product attention to capture the contextual relationship between k d-dimensional patches, resulting in the computational complexity of $O(k^2)$. By contrast, Separable Self-Attention [36] computes contextual scores relative to one latent patch, which are then used to reweight the input patches and generate a context vector that encodes the global information. Since these scores are computed with respect to a single patch instead of k patches, Separable Self-Attention reduces the computational complexity by a factor k; it also allows us to encode the global information by replacing the quadratic MHSA with two separate linear operations, finally decreasing the complexity from $O(k^2)$ to $O(k)$.

A diagram of Separable Self-Attention is shown in Figure 3(b). Given the input signal x, the entire attention process can be accordingly described as follows:

$$y = \left(\underbrace{\sum \left(\overbrace{\sigma(xW_I)}^{c_s \in \mathbb{R}^k} * xW_K \right)}_{c_v \in \mathbb{R}^d} * ReLU(xW_V) \right) W_O, \tag{1}$$

where W_I, W_K, W_V, and W_O are weights of four corresponding linear layers. σ is the softmax function. c_s is the context score projected from x via the linear layer with weight W_I. The context vector c_v is calculated using W_I and W_K, which is essentially a cheap version of the attention matrix and shared across all patches in x. The contextual information in c_v is then propagated by a broadcast element multiplication operation denoted as $*$ in Equation (1).

Spatial Attention. To reassign features passing through Fast Transformer in the spatial domain, Spatial Attention [55] is applied at the end of each Fast Transformer. As described in Figure 3(c), Spatial Attention adopts an adaptive spatial region selection mechanism: given input feature maps x, max pooling and average pooling operations are applied across channels to produce two attention masks, which are then concatenated and processed by a convolution with the size of $9 \times 9 \times 1$ to evaluate the contribution of each spatial position through a Sigmoid layer.

Channel Attention. Furthermore, we embed simplified Channel Attention after three Fast Transformers. Unlike previous works [5, 19], we simply multiply output channels by a set of learned parameters $\gamma, \gamma \in \mathbb{R}^{C\times1\times1}$, one γ value for a channel. The magnitude of γ is not constrained. In this way, each channel is weighted independently and the network is able to spontaneously assign different weights to channels. As such, we can maximize the use of inter-channel information.

4 META-LEARNING-BASED POST-PROCESSING

4.1 The Meta-PP Framework

Previous learning-based filtering models sometimes suffer from a large gap between the training set and the test set because most models are trained offline and thus fail to make adjustments when encountering unseen content. This can be effectively addressed by incorporating external and internal learning, because in this way, (1) a strong and stable mapping relationship can be established based on large-scale external datasets, and (2) the internal prior of the image itself can be exploited and utilized. In light of this, we propose an adaptive post-processing framework using Meta-learning to explore both external datasets and internal data information for better post-processing performance.

Specifically, we utilize the prominent Model-Agnostic Meta-learning [11], where the adaptation process is simulated by gradient updates to weighting parameters. Model-Agnostic Meta-learning first samples a set of examples from the current task $T_i \sim p(T)$, where $p(T)$ represents the distribution of tasks. Then, the current task is adapted by fine-tuning the model parameters with **Stochastic Gradient Descent (SGD)** [11]. Inner loop training and outer loop training take turns to work. The inner loop step is applied with the purpose of learning the target task, described as follows:

$$\theta_i^{'} = \theta - \alpha \nabla_\theta \mathcal{L}_{T_i}^{in}(f_\theta), \tag{2}$$

where θ denotes a collection of weighting parameters.

Afterward, the weighting parameters θ can be further updated by the outer loop:

$$\theta \leftarrow \theta - \beta \nabla_\theta \sum_{T_i \sim p(T)} \mathcal{L}_{T_i}^{out}(f_{\theta_i}^{'}), \tag{3}$$

with α, β being the learning rate of the inner and outer loops and f_θ being our backbone model. $\mathcal{L}_{T_i}^{in}(\cdot)$ and $\mathcal{L}_{T_i}^{out}(\cdot)$ refer to the loss to be optimized. The inner loop and outer loop work alternatively to seek network initialization parameters that can best satisfy all tasks, and allow a small fraction of these parameters to be fine-tuned quickly and easily. When the training is completed, the test is performed with untrained data. Note that only the inner loop will be conducted in the testing phase. At this time, small local changes in parameters can lead to a huge improvement in the performance of the neural model.

4.2 Compression Degradation Estimation

In contrast to the in-loop filtering problem where the ground truth frames are known, the post-processing task knows nothing about the original frames—that is, the internal compression distortion of each frame is unavailable, making it impossible to update the neural model on the fly. One

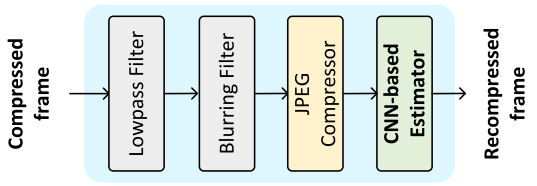


Fig. 4. Architecture of the proposed CDE, which consists of four parts: a lowpass filter, a blurring filter, a JPEG compressor, and a CNN-based estimator.

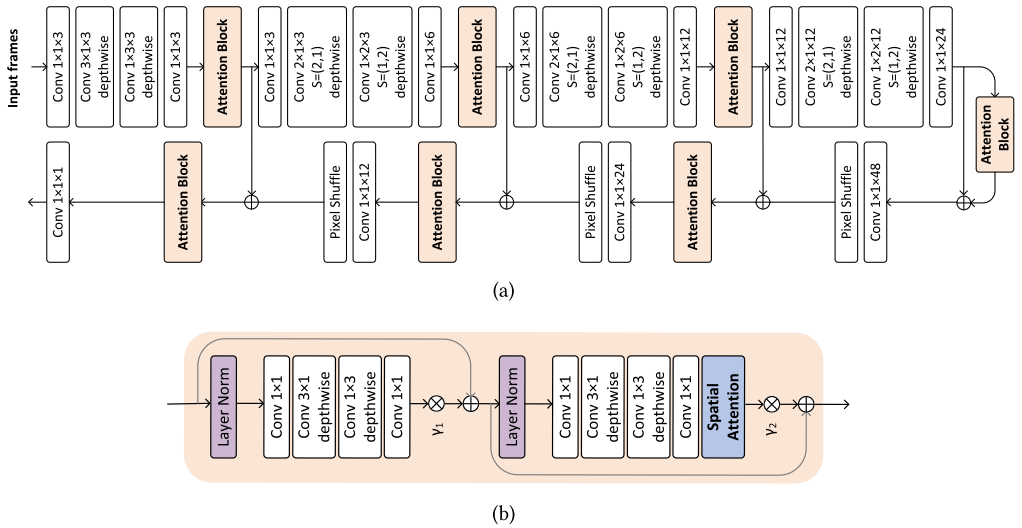


Fig. 5. The CNN-based estimator used in the proposed CDE model. (a) Overall structure of the proposed CNN-based estimator. (b) Details of the Attention Block that uses Spatial Attention and Channel Attention for feature collection.

straightforward solution is to compress the current reconstructed frame again to learn inherent distortion clues. However, it is usually computationally expensive and time consuming to call the video codec to recompress the current frame. To tackle this problem, we propose the CDE model to estimate the distortion caused by the video codec. Using information provided by CDE, we can update BackboneFormer for improved results. Incorporating CDE and BackboneFormer into the Meta-learning framework, we develop a Meta-PP solution, as described in the following.

The classical degradation model [52] generally includes the blur filter, noise generator, rescaling, and JPEG/FFMPEG compressor, among others. It is hard for these basic operators to cover all degradation cases, especially those caused by complex video compression. To address this limitation, the proposed CDE combines the learning-based network and classical basic operators together to model the degradation procedure. As such, the classical operators provide a stable estimation and the neural network further improves the accuracy of the estimation using its powerful nonlinear capability. As depicted in Figure 4, CDE consists of a lowpass filter, a blurring filter, a JPEG compressor, and a CNN-based estimator.

To reduce the computational complexity and time consumption, we construct the CNN-based estimator following the U-Net style, as illustrated in Figure 5. The entire network is built on the *multi-scale encoder-decoder architecture*: the encoder downsamples the input frame three times and

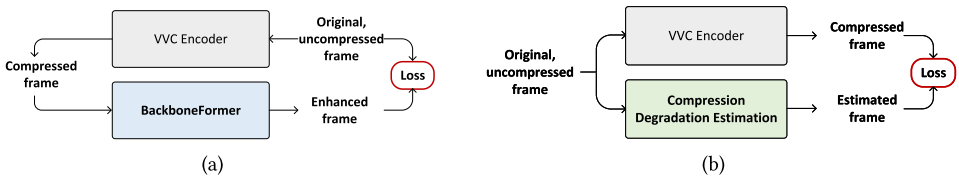


Fig. 6. Pre-training of the proposed BackboneFormer and the CDE model. (a) The pre-training method for the BackboneFormer network. (b) The pre-training method for the CDE model.

produces latent features, the decoder symmetrically upsamples the latent features and outputs processed frames, and skip connections pass through features from the encoder to the decoder for multi-scale features fusion. The structure of U-Net naturally fits the idea of the degradation model by projecting input features into a high-dimensional space and then reconstructing from it, with skip connections further supplementing high-frequency information. In this way, we can effectively drop trivial information and disturb the uniform distortion caused by these basic operators. Moreover, to increase the adaptive capacity of the CNN-based estimator, we embed the Attention Block in each scale, as illustrated in Figure 5. Specifically, the Attention Block adopts only Spatial and Channel Attention to reduce computational complexity.

4.3 Meta-PP Training

We train the proposed Meta-PP in two steps: first, BackboneFormer and CDE are *pre-trained* on external datasets separately; then, these two pre-trained models are involved in the Meta-learning process for iterative updating, where *outer loop training* and *inner loop training* take turns to fine-tune model parameters.

Pre-Training. Figure 6 illustrates the pre-training process of these two networks. Without loss of generality, we use the $\mathcal{L}_2$ loss function for minimum MSE optimization between the generated frame and the label frame. To train the CDE model, as illustrated in Figure 6(b), the original, uncompressed frame goes across the VVC encoder and the CDE module in parallel, generating the compressed frame and the estimated frame, respectively. The training then targets minimizing the MSE of these two generated frames until model convergence. To train BackboneFormer, as illustrated in Figure 6(a), the compressed frame is sent to BackboneFormer for the generation of an enhanced frame. The training then targets minimizing the MSE between the enhanced frame and the original frame.

Outer Loop Training. The outer loop training (Figure 7(a)) aims to make the BackboneFormer model more flexible and intelligent to various video characteristics and to polish a more accurate CDE model for the current input frame. In this regard, we define two $\mathcal{L}_1$ loss functions for the two networks:

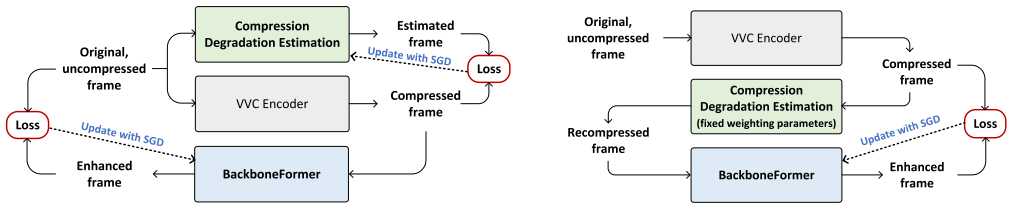
$$\mathcal{L}_{CDE}^{out} = ||CDE(\mathbf{I}, \phi) - \mathbf{I}_{rec}||_1 \tag{4}$$

and

$$\mathcal{L}_{BBF}^{out} = ||BBF(\mathbf{I}_{rec}, \theta) - \mathbf{I}||_1, \tag{5}$$

where $CDE(\cdot)$ denotes the CDE model with weighting parameters ϕ and $BBF(\cdot)$ denotes the BackboneFormer model with weighting parameters θ. Here, each loss updates the corresponding network parameters separately. Note that the loss is calculated with updated parameters θ' (updated from θ) and parameters ϕ (not updated), yet the gradient is still computed using θ and ϕ.

Inner Loop Training. In each epoch of Meta-training, we apply the outer loop training first and then the inner loop training. As illustrated in Figure 7(b), for the inner loop training, we fix



(a) Outer loop training (b) Inner loop training

Fig. 7. Meta-PP training for BackboneFormer and CDE, which applies the outer loop and inner loop training alternatively to fine-tune model parameters. (a) Outer loop training for updating two models. (b) Inner loop training for updating BackboneFormer only.

the CDE parameters and feed the compressed frame into CDE. As such, the recompressed frame is produced and immediately sent into BackboneFormer for inference, producing the enhanced frame. The loss function $\mathcal{L}_{BBF}^{in}$ is then applied to minimize the distance between this enhanced frame and the compressed frame:

$$\mathcal{L}_{BBF}^{in} = ||BBF(CDE(\mathbf{I}_{rec}, \phi), \theta) - \mathbf{I}_{rec}||_1, \tag{6}$$

where weighting parameters ϕ of the CDE model are fixed and only θ is updated. The entire Meta-PP training process is described in Algorithm 1.

ALGORITHM 1: Meta-PP Training

Require: $p(T)$: uniform distribution over frames
Require: α, β: inner/outer-loop learning rates
1: Initialization: pre-trained parameters θ and ϕ
2: **while** Meta-PP training **do**
3: Sample a batch of frames $T_i \leftarrow p(T)$
4: **for** i **do**
5: Get $\mathbf{I}$ and $\mathbf{I}_{rec}$ from T_i
6: Generate $\mathbf{I}_{recom}$ using CDE_ϕ
7: Calculate $\nabla_{\phi,\theta}, \mathcal{L}^{in}$ according to Equation (2)
8: Compute adapted weighting parameters $\theta^{'}$: $\theta^{'} = \theta - \alpha\nabla_\theta \mathcal{L}_{BBF}^{in}$
9: **end for**
10: Update $\phi \leftarrow \phi - \beta\nabla_\phi \sum_{T_i} \mathcal{L}_{CDE}^{out}$
11: Update $\theta \leftarrow \theta - \beta\nabla_\theta \sum_{T_i} \mathcal{L}_{BBF}^{out}$
12: **end while**

4.4 Meta-PP Testing

In the test, the inner loop update is performed only once to adapt BackboneFormer to the input frame. As depicted in Figure 8, the recompressed frame is estimated directly by the CDE model with fixed weighting parameters; this recompressed frame is preliminary enhanced by BackboneFormer and compared with the previously compressed frame for loss calculation, which in return drives the updating of BackboneFormer model through the SGD algorithm. As such, BackboneFormer elegantly adjusts itself to the new input content variations. Once updated, BackboneFormer inputs the low-quality compressed frame again for enhancement, producing a high-quality frame. The entire Meta-PP testing process is expressed in Algorithm 2.

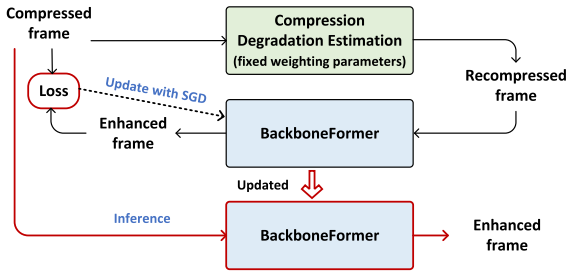


Fig. 8. Meta-PP testing where weighting parameters of the CDE model are fixed to update BackboneFormer only. The updated BackboneFormer is used to enhance the compressed frame.

ALGORITHM 2: Meta-PP Testing

Require: $\mathbf{I}_{rec}$: input compressed image
Require: μ: update step size in test
Require: CDE_ϕ with fixed weighting parameters
1: Initialize θ with pre-trained parameters
2: Generate $\mathcal{L}_{recom}$ using CDE_ϕ
3: Compute adapted weighting parameters with SGD:
$\theta \leftarrow \theta - \mu \nabla_\theta \mathcal{L}(BBF_\theta(CDE_\phi(\mathbf{I}_{rec})), \mathbf{I}_{rec})$
4: Compute $BBF_\theta(\mathbf{I}_{rec})$

To further reduce the testing complexity, we use first-order approximation to implement the SGD calculated with the second derivation in Equation (2), which has been proven to yield competitive results with lower computational cost [11, 39].

5 EXPERIMENTAL RESULTS

5.1 Experiment Settings

Training Configurations. To train our proposed network, we first build our training samples using the DIV2K [23] dataset. All images in DIV2K are converted to YUV format as uncompressed raw frames. These images are then compressed using VVC reference software VTM-15.0 under All Intra configuration. The reconstructed frames are collected as samples for training.

In the training, we adopt a progressive training strategy [61], which gradually goes from large batch size with small patch size to small batch size with large patch size. In this way, the network can be trained stably and benefit from a large patch size, finally learning better contextual information compared to the conventional training method with a fixed batch or patch size. Specifically, we initially crop the training sample randomly to 64×64 patches with batch size 64 and then apply the progressive training strategy with patch size and batch size set to $\{128 \times 128, 32\}$, $\{256 \times 256, 16\}$, and $\{512 \times 512, 8\}$, sequentially. We separately train four models for QP values 22, 27, 32, and 37. The Adam solver [27] is used as the optimizer with an initial learning rate of 1e-3 and decayed every 100 epochs with a factor of 0.5. Each time we change the patch size and batch size, we set the learning rate to 1e-4 from scratch. Except for the 64×64 patch size being trained for 200 epochs, all other patch sizes are trained for 150 epochs only.

Test Configurations. The 18 test sequences selected by the Joint Collaborative Team on Video Coding (JCT-VC) for video codec testing and quality evaluation are used for testing. Note that these sequences are excluded from our training dataset. We strictly follow the common test conditions of

Table 1. BD-Rate Comparison with State-of-the-Art Methods with VVC VTM-15.0 Serving as the Anchor

Resolution	Sequence	VRCNN [6]	SEFCNN [7]	LCMMCNN [49]	NRCNN [31]	NRCNN + Meta-PP	BBF	BBF + Meta-PP	BBF Ext.	BBF Ext. + Meta-PP
2K	PeopleOnStreet	−0.98%	−2.13%	−2.89%	−4.18%	−5.43%	−3.15%	−3.98%	−5.23%	−6.73%
	Traffic	−1.84%	−3.62%	−4.21%	−6.03%	−6.09%	−3.05%	−3.50%	−6.01%	−5.72%
1080p	BasketballDrive	−0.68%	−2.59%	−4.91%	−7.15%	−7.24%	−2.31%	−3.25%	−6.75%	−7.54%
	BQTerrace	−0.59%	−1.57%	−2.42%	−3.66%	−5.09%	−2.75%	−3.74%	−4.89%	−5.93%
	Cactus	−0.44%	−1.17%	−3.09%	−4.92%	−5.48%	−1.86%	−2.70%	−5.80%	−5.22%
	Kimono	−1.38%	−3.07%	−4.48%	−6.36%	−5.79%	−2.79%	−3.50%	−7.07%	−6.66%
	ParkScene	−1.92%	−3.32%	−4.34%	−6.01%	−6.02%	−3.13%	−3.83%	−6.66%	−6.57%
WVGA	BasketballDrill	−2.87%	−6.41%	−6.76%	−9.40%	−8.64%	−5.04%	−6.04%	−9.18%	−10.84%
	BQMall	−2.57%	−4.82%	−5.01%	−7.11%	−8.45%	−4.36%	−4.96%	−7.19%	−7.42%
	PartyScene	−1.80%	−2.99%	−3.39%	−4.65%	−5.69%	−3.17%	−3.73%	−5.23%	−4.81%
	RaceHorsesC	−0.74%	−1.63%	−2.37%	−3.37%	−4.72%	−2.08%	−2.39%	−4.18%	−4.61%
WQVGA	BasketballPass	−3.04%	−5.75%	−5.91%	−8.37%	−9.15%	−5.58%	−5.50%	−9.17%	9.43%
	BlowingBubbles	−2.30%	−3.82%	−4.36%	−5.98%	−7.69%	−3.79%	−4.45%	−6.78%	−7.83%
	BQSquare	−3.00%	−5.10%	−5.27%	−6.89%	−8.25%	−5.14%	−6.78%	−8.25%	−10.05%
	RaceHorses	−2.73%	−4.14%	−4.62%	−5.98%	−6.59%	−3.83%	−4.24%	−6.21%	−6.37%
720p	FourPeople	−2.58%	−5.21%	−5.27%	−7.73%	−7.58%	−4.74%	−5.38%	−7.76%	−8.18%
	Johnny	−2.04%	−4.72%	−5.16%	−7.89%	−8.95%	−4.20%	−5.01%	−8.09%	−8.41%
	KristenAndSara	−2.17%	−4.58%	−5.01%	−7.21%	−7.79%	−4.00%	−4.72%	−7.33%	−7.57%
	Average	−1.87%	−3.70%	−4.41%	−6.27%	−6.92%	−3.61%	−4.32%	−6.77%	−7.22%

BackboneFormer is written as BBF for short.

VVC to collect R-D performance at QPs 22, 27, 32, and 37. The trained models are deployed on the VTM-15.0 platform for evaluation. We invoke Python to call the trained models. The BD-Rate [2] is used to evaluate the compression efficiency.

Our model prototype is implemented using the PyTorch platform. Our training and testing are performed on a computer with an Intel i7-9700K CPU at 3.6 GHz, with 64 GB of main memory and a 3090 GPU. The runtime and MACs cost are also tested in this environment.

5.2 Comparison to State-of-the-Art Methods

We compare the performance of our proposed method with representative learning-based filtering methods, including VRCNN [6], SEFCNN [7], LCMMCNN [49], and NRCNN [31].

Table 1 reports the BD-Rate reduction of different methods against the VVC anchor. Table 2 reports their computational complexity. As can be seen, with a comparable number of parameters, the proposed BackboneFormer attains a much higher BD-Rate reduction than VRCNN (i.e., 3.61% vs. 1.87%). Using the Meta-PP framework (i.e., "BackboneFormer + Meta-PP" in Table 1) further improves the enhancement performance from 3.61% to 4.32%, which outperforms SEFCNN and is comparable to LCMMCNN. Notice that SEFCNN has 596K parameters and LCMMCNN has 370K, which is about 12× and 7× that of the proposed BackboneFormer. Clearly, the proposed BackboneFormer aggregates and embeds information more efficiently and effectively. NRCNN obtains higher BD-Rate gains than ours (i.e., 6.27% vs. 4.32%) because it involves a much larger model using 1,515K parameters. For a fair comparison, we simply extend BackboneFormer to a large model having 1,500K parameters. As shown in the last two columns of Table 1, using the same number of parameters, the proposed "BackboneFormer Ext." attains 6.77% and "BackboneFormer Ext. + Meta-PP" attains as high as 7.22% BD-Rate reduction, which significantly surpasses the 6.27% of NRCNN.

Furthermore, we incorporate NRCNN into the proposed Meta-PP framework for performance evaluation. As reported in Table 1, the BD-Rate reduction is improved from 6.27% to 6.92%, obtaining an additional gain of 0.65%. This demonstrates the superiority and generalizability of our method.

Table 2. Computational Complexity Comparison with State-of-the-Art Methods

Method	VRCNN [6]	SEFCNN [7]	LCMMCNN [49]	NRCNN [31]	BackboneFormer	BackboneFormer + Meta-PP
Parameters	54K	596K	370K	1,515K	50K	61K
MACs (per pixel)	55K	484K	316K	1,517K	49K	50K
Processing time (ms per block)	6.95	34.90	19.94	73.80	32.21	86.71

(a) VVC compressed (b) BackboneFormer (c) BackboneFormer + Meta-PP (d) Original

Fig. 9. Visualization comparison. The first line is from the sequence *BasketballDrill* and the second line is from the sequence *FourPeople*.

5.3 Visualization

We visualize the VVC compressed frame, the enhanced frame, and the original frame in Figure 9 for qualitative comparison. Overall, the VVC in-loop filters remove most compression artifacts, but the reconstructed frames still contain some blocking and ringing noise, such as the basketball net and the English words shown in Figure 9(a). The proposed method successfully removes these artifacts, exhibiting very clear edges and legible characters (see Figure 9(b) and (c)). As a result, the enhanced frames look visually pleasant and pretty close to the original, uncompressed frames.

5.4 Complexity

We compare the computational complexity of different methods in Table 2, including the model parameters and MACs (per pixel). As seen, our model consumes the smallest number of parameters and MACs of all: for the model parameters, our BackboneFormer has only 50K parameters, and "BackboneFormer + Meta-PP" has 61K; concerning the MACs, BackboneFormer costs 49k/pixel, and "BackboneFormer + Meta-PP" costs 50K (the additional 1K MACs/pixel is introduced by the CDE model), which is much lower than other methods. NRCNN obtains 6.27% BD-Rate gains at the expense of as high as 1,517K MACs/pixel, which is 30× that of BackboneFormer.

Moreover, we report the processing time of each method. Notice that this time is investigated on the GPU platform because we invoke Python for post-processing. For a fair comparison, we report the average time to process a 64×64 block on the same GPU. Results show that the processing time is related to the model size: for most methods, the larger the model, the longer the processing

Table 3. Time Complexity (Seconds Per Frame) Comparison with the VVC Anchor

Resolution	VVC Anchor	W/ BackboneFormer	W/ BackboneFormer + Meta-PP
2K	0.27	134%	453%
1080p	0.12	197%	217%
WVGA	0.03	465%	560%
WQVGA	0.01	627%	1,457%
720p	0.04	419%	829%
Average	0.08	382%	697%

time. We also notice that the proposed method costs as long as 32.21-ms processing time. This occurs because the Transformer module used in BackboneFormer involves matrix calculation, which is much more complex than conventional convolution operations used in other methods. We believe that our processing time will be improved in the future as the experimental platform adopts dedicated techniques to accelerate the Transformer and model inference. For example, in practical implementations, TensorRT,[1] an SDK for high-performance deep learning inference, can be used for acceleration through the use of operator fusion, quantization, and other methods. However, the complex matrix calculation in the Transformer can be simplified by average pooling operations at the expense of slight performance degradation [48, 60]. In addition, due to the additional parameter update iteration in inference, our "BackboneFormer + Meta-PP" costs a longer processing time (86.71 ms). As observed, the proposed method exhibits a limitation in terms of processing time, which will be deferred to our future study.

We also investigate the average runtime (seconds per frame) of each method when deploying the proposed method on the VVC codec. The runtime is derived when the VVC decoder is set at QP value 37 under the All Intra configuration. We compare the runtime of the VVC decoder and the proposed Meta-PP in Table 3. Defining the decoding time of the VVC anchor as T_{anchor}, we report the runtime of BackboneFormer following

$$T_{complexity} = \frac{T_{BackboneFormer}}{T_{anchor}} \times 100\%. \tag{7}$$

On average, the VVC decoder consumes 0.08 s/frame for decoding videos of different resolutions. With the proposed BackboneFormer, the decoding time is increased by 382% compared to the VVC anchor. As aforementioned, the extra running time is mainly owing to the complicated self-attention operations of BackboneFormer. The use of "BackboneFormer + Meta-PP" needs a longer time (i.e., 697% against the VVC anchor) since an additional round of parameter updating is required.

5.5 Ablation Study

This section conducts a series of experiments to study the contribution of each component in our proposed framework.

Transformer Dissection. The core of BackboneFormer is the Separable Self-Attention mechanism. In Table 4, we replace the Separable Self-Attention module (called *Separable SA*) with the conventional MHSA in Figure 3(a), the widely used Window-based MHSA (W-MSA) [33], and the Stochastic Window-based MHSA (StoWin-MSA) [56]. At QP value 37, we test their average coding improvements over the VVC anchor and their processing time on GPU for each frame.

[1] https://developer.nvidia.com/tensorrt

Table 4. Average PSNR (dB) Gain and Processing Time (s) of BackboneFormer Using Different Attention Mechanisms

Resolution	Sequence	MHSA [10]	W-MSA [33]	StoWin-MSA [56]	Separable SA
2K	PeopleOnStreet	0.0603 / 71.4399	0.1967 / 0.0521	0.2410 / 53.4962	0.2707 / 0.0229
	Traffic	0.0325 / 69.9928	0.1941 / 0.0519	0.2340 / 51.3808	0.2227 / 0.0260
1080p	BasketballDrive	0.0955 / 36.4317	0.0951 / 0.0389	0.1328 / 26.2869	0.1273 / 0.0246
	BQTerrace	0.0977 / 36.2529	0.2561 / 0.0379	0.2693 / 26.3323	0.2594 / 0.0239
	Cactus	0.0496 / 36.3248	0.1526 / 0.0369	0.1888 / 26.3043	0.1465 / 0.0259
	Kimono	0.0301 / 36.3996	0.1269 / 0.0389	0.1639 / 26.3218	0.1580 / 0.0249
	ParkScene	0.0227 / 36.3397	0.1489 / 0.0379	0.2049 / 26.3241	0.1695 / 0.0239
WVGA	BasketballDrill	0.0554 / 7.5252	0.2830 / 0.0379	0.3115 / 5.4146	0.3302 / 0.0239
	BQMall	0.0659 / 7.5277	0.2931 / 0.0379	0.3302 / 5.3984	0.3384 / 0.0240
	PartyScene	0.0647 / 7.5254	0.3330 / 0.0379	0.3687 / 5.3761	0.3938 / 0.0239
	RaceHorsesC	0.0811 / 7.5250	0.1535 / 0.0359	0.1873 / 5.4102	0.1547 / 0.0239
WQVGA	BasketballPass	0.0255 / 2.1472	0.3628 / 0.0419	0.3827 / 1.6911	0.4568 / 0.0249
	BlowingBubbles	0.0375 / 2.1511	0.2634 / 0.0379	0.2961 / 1.6818	0.3150 / 0.0229
	BQSquare	0.0177 / 2.1493	0.7116 / 0.0359	0.7306 / 1.6904	0.8583 / 0.0249
	RaceHorses	0.0100 / 2.1448	0.2791 / 0.0379	0.3163 / 1.6866	0.3050 / 0.0239
720p	FourPeople	0.0386 / 16.1222	0.3437 / 0.0369	0.3605 / 11.9300	0.4147 / 0.0239
	Johnny	0.1754 / 16.1170	0.2605 / 0.0359	0.2992 / 11.9165	0.2682 / 0.0269
	KristenAndSara	0.0863 / 16.1231	0.3030 / 0.0349	0.3288 / 11.9595	0.3327 / 0.0239
	Average	0.0582 / 22.7911	0.2643 / 0.0392	0.2970 / 16.7001	0.3068 / 0.0244

In the table, the two numbers denote ΔPSNR and processing time.

Theoretically, MHSA promises higher PSNR gains than Separable SA due to its ability to capture more correlations between patches. However, this advantage is largely diminished in implementation because of its high complexity: the training and testing patch size is limited to 128×128 in our experimental platform, hindering proper model training and global information exploration. By contrast, Separable SA has much lower computational complexity and thus enjoys a larger patch size and receptive field. As a result, using Separable SA yields a much higher PSNR gain and drastically reduced processing time compared to the use of MHSA. On average, our Separable SA provides a 0.3068-dB PSNR gain with an average processing time of 0.0244 seconds versus the MHSA's 0.0582-dB gain with 22.7911 seconds (1,000× slower than Separable SA). W-MSA achieves slightly lower performance than Separable SA but with a longer processing time. As for StoWin-MSA, it performs comparably with Separable SA (0.2970 dB vs. 0.3068 dB); however, it needs much more time due to its cumbersome layer expectation propagation algorithm.

Compression Distortion Estimation. Table 1 shows the effectiveness of Meta-PP. It can be seen that by introducing a CDE model for distortion estimation, the BD-Rate reduction improves from 3.61% to 4.32%. The CDE model has a very small computational cost, which only involves 11K parameters and 1K MACs/pixel. To further verify the performance of CDE, we conduct an ablation study using another distortion estimator—that is, the JPEG estimator [58]. As reported in Table 5, the JPEG estimator leads to 3.85% BD-Rate reduction, which is slightly better than BackboneFormer itself. Nevertheless, when using the proposed CDE, the BD-Rate gains can be as high as 4.32%, indicating that CDE can better model the compression distortion for more accurate estimation.

We additionally compare the JPEG-estimated frame, the CDE-estimated compressed frame, and the VVC real compressed frame, as visualized in Figure 10. As well, the JPEG-estimated recompressed frame, the CDE-estimated recompressed frame, and the VVC real recompressed counterpart are shown. It is observed that the estimated and real compressed frames look quite close to each other, whereas the JPEG-estimated frame contains more artifacts and distortion. We

Table 5. Ablation Study on the CDE Model

Resolution	Sequence	JPEG Estimator [58]	Proposed CDE
2K	PeopleOnStreet	−3.59%	−3.98%
	Traffic	−3.10%	−3.50%
1080p	BasketballDrive	−2.48%	−3.25%
	BQTerrace	−3.29%	−3.74%
	Cactus	−2.19%	−2.70%
	Kimono	−2.85%	−3.50%
	ParkScene	−3.35%	−3.83%
WVGA	BasketballDrill	−6.44%	−6.04%
	BQMall	−4.53%	−4.56%
	PartyScene	−3.43%	−3.73%
	RaceHorsesC	−2.05%	−2.39%
WQVGA	BasketballPass	−4.19%	−5.50%
	BlowingBubbles	−4.03%	−4.45%
	BQSquare	−6.10%	−6.78%
	RaceHorses	−3.90%	−4.24%
720p	FourPeople	−5.00%	−5.38%
	Johnny	−4.42%	−5.01%
	KristenAndSara	−4.26%	−4.72%
	Average	−3.85%	−4.32%

also calculate the MSE and SSIM metrics of these frames against their original frames. As seen in Figure 10, JPEG-estimated frames have the highest MSE and lowest SSIM values of all; the CDE-estimated frames have comparable MSE with the VVC real compressed frames and their SSIM values are also quite close—for example, for sequence *Johnny*, their SSIM values are both beyond 0.90, demonstrating their high similarity to each other.

Attention Mechanism. Table 6 reports ablation studies on the Spatial Attention and Channel Attention modules. As can be seen, without both Spatial and Channel Attention, the coding performance of BackboneFormer decreases from 3.61% to 3.02%. Disabling Spatial/Channel Attention gains only 3.21%/3.36%, which are lower than the proposed BackboneFormer. Furthermore, we show in Figure 11 the attention maps yielded by Spatial Attention. In addition, we visualize the feature maps without and with Channel Attention in Figure 12. Clearly, higher attention is assigned to edges and textures in a frame when using Spatial or Channel Attention.

The Number of Update Iterations. The number of update iterations also impacts post-processing performance. Figure 13 illustrates the BD-Rate gains when using different numbers of update iterations in testing. As can be seen, increasing the number of iterations leads to higher BD-Rate gains, as the model better adapts itself to the current frame. However, this comes up with an unacceptable increase in processing time. As demonstrated in Table 2, even using a single-time iteration extends the processing time from 32.21 to 86.71 ms, not to mention the multiple-time iterations. Therefore, in this work, we adopt one iteration in the model inference to achieve a tradeoff between the processing time and enhancement performance.

5.6 Discussion

Overall, the proposed method performs content-aware post-processing from three aspects: (1) the BackboneFormer model uses Fast Transformer, which incorporates multiple attention components

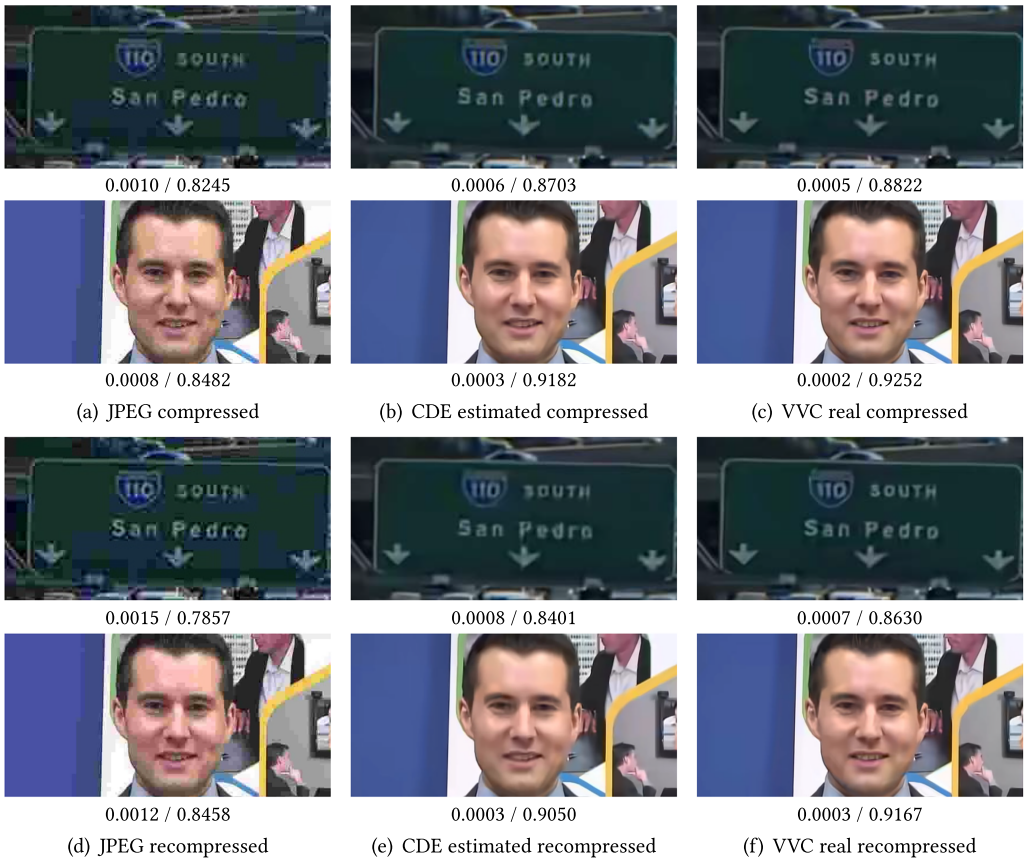


Fig. 10. Visualization of CDE estimated and VVC compressed frames. MSE/SSIM values are also reported to demonstrate their similarity to the original frames.

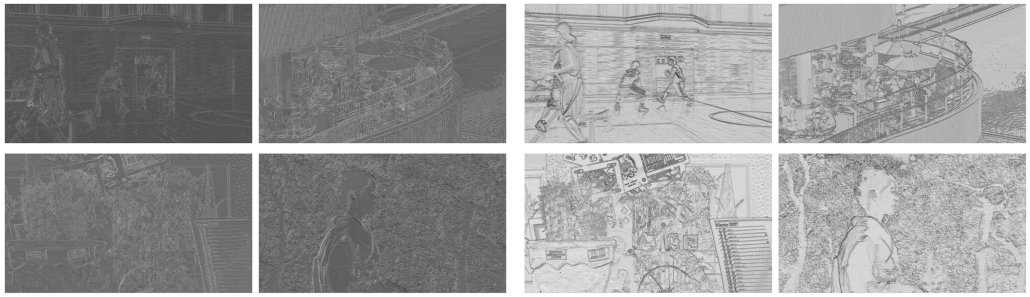
to perceive and aggregate information, providing a reliable foundation for undertaking filtering tasks; (2) the CDE model leverages both classical operators and the neural model to best approximate the compression distortion; and (3) the Meta-PP framework further refines the pre-trained BackboneFormer model according to the estimated distortion from CDE. As can be seen in Table 1, BackboneFormer itself already achieves significant gains over the VVC anchor, exhibiting its capability for content-aware processing. The integration of Meta-PP further improves the BD-Rate performance, demonstrating its superiority in instantaneous adaption to dynamic video content.

Recalling our motivation presented in Section 1, we aim to adapt statistically learned models to dynamic video content, which is associated with two tasks in video coding: in-loop filtering and post-processing. These two tasks are similar yet differ in their standpoints, as the in-loop filtering task is applied in both the encoder and decoder while the post-processing task is applied optionally in the decoder. To efficiently and effectively fulfill these two tasks, we suggest distinct solutions. For in-loop filtering, we utilize side information abstracted from the original video to guide the neural model, as introduced in our prior work [8, 28]. In contrast, the post-processing task poses more difficulty since the original signal is unknown. To address this challenge, we leverage a Meta-learning framework and devise a CDE algorithm to update the model weighting parameters, dynamically adapting the neural model to incoming videos.

Table 6. Ablation Study on the Spatial Attention and Channel Attention Modules

Resolution	Sequence	W/o Spatial & Channel Attention	W/o Spatial Attention	W/o Channel Attention	BackboneFormer
2K	PeopleOnStreet	−2.81%	−3.00%	−3.25%	−3.15%
	Traffic	−2.66%	−2.87%	−2.98%	−3.05%
1080p	BasketballDrive	−2.16%	−3.06%	−2.91%	−2.31%
	BQTerrace	−2.53%	−2.59%	−2.88%	−2.75%
	Cactus	−2.03%	−2.26%	−2.31%	−1.86%
	Kimono	−2.54%	−3.18%	−3.05%	−2.79%
	ParkScene	−3.21%	−3.56%	−3.48%	−3.13%
WVGA	BasketballDrill	−3.82%	−4.05%	−3.71%	−5.04%
	BQMall	−3.20%	−3.18%	−3.63%	−4.36%
	PartyScene	−2.63%	−2.77%	−3.06%	−3.17%
	RaceHorsesC	−1.45%	−1.67%	−1.82%	−2.08%
WQVGA	BasketballPass	−4.15%	−3.67%	−4.09%	−5.58%
	BlowingBubbles	−3.25%	−3.43%	−3.26%	−3.79%
	BQSquare	−4.32%	−3.99%	−4.23%	−5.14%
	RaceHorses	−3.23%	−3.42%	−3.24%	−3.83%
720p	FourPeople	−3.43%	−3.66%	−4.40%	−4.74%
	Johnny	−3.56%	−3.87%	−4.27%	−4.20%
	KristenAndSara	−3.30%	−3.60%	−3.89%	−4.00%
	Average	−3.02%	−3.21%	−3.36%	−3.61%

Fig. 11. Attention maps from Spatial Attention. The red color implies higher attention.



(a) W/o Channel Attention (b) W/o Channel Attention

Fig. 12. Visualization of feature maps with and without Channel Attention. All output feature maps are averaged for visualization.

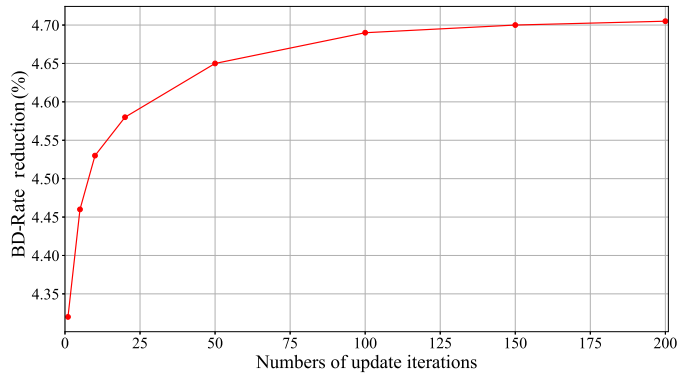


Fig. 13. Enhancement performance versus iteration numbers in testing. Different numbers of update iterations are applied in testing to show the BD-Rate gains. The gain curve converges when the iteration number reaches 150.

Notice that we use Meta-learning for post-processing rather than in-loop filtering because the SGD used in Meta-learning may produce varying models after each update, making it unsuitable for the in-loop filtering task that requires the encoder and decoder to share the same neural model for matching.

6 CONCLUSION

This article presented a content-aware post-processing approach to improve the quality of compressed videos. We devised BackboneFormer using stacked Fast Transformers to aggregate and embed latent features. Each Fast Transformer contains Separable Self-Attention, Spatial Attention, and Channel Attention for dynamic weight adjustment according to the importance of input content. Furthermore, we developed a Meta-PP framework to update the weighting parameters of the BackboneFormer model on the fly. In this way, the statistically learned neural model can adapt to dynamic input content for better performance. Since the original videos are unavailable in post-processing (i.e., the compression distortion is unknown), we hence proposed a CDE model, which incorporates both classic operators and a neural network, to approximate the compression distortion. The approximated distortion is then utilized in Meta-PP training and testing for BackboneFormer updating. Experimental results showed that a small-scale BackboneFormer with only 50K parameters attains 3.61% BD-Rate reduction against the VVC anchor. Further, the Meta-PP improves BD-Rate gains to 4.32% with low computational complexity, confirming its superiority.

In our previous work, we developed an MSR (Multi-hypothesis Sample Refinement) approach for in-loop filtering. We are continuing to study how to collaboratively utilize in-loop MSR and out-loop Meta-PP in the video coding framework for better adaptation to various video dynamics. Currently, the proposed BackboneFormer and Meta-PP take longer processing time than that of conventional neural models. It is our future work to reduce the processing time while maintaining the enhancement performance.

REFERENCES

[1] Yuval Bahat, Netalee Efrat, and Michal Irani. 2017. Non-uniform blind deblurring by reblurring. In *Proceedings of the IEEE International Conference on Computer Vision*. 3286–3294.

[2] G. Bjøntegaard. 2001. Calculation of average PSNR differences between RD-curves. In *Proceedings of the 13th Meeting of the Video Coding Experts Group*.

[3] Hanting Chen, Yunhe Wang, Tianyu Guo, Chang Xu, Yiping Deng, Zhenhua Liu, Siwei Ma, Chunjing Xu, Chao Xu, and Wen Gao. 2021. Pre-trained image processing transformer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 12299–12310.

[4] Haibo Chen, Lei Zhao, Zhizhong Wang, Huiming Zhang, Zhiwen Zuo, Ailin Li, Wei Xing, and Dongming Lu. 2021. Artistic style transfer with internal-external learning and contrastive learning. *Advances in Neural Information Processing Systems* 34 (2021), 26561–26573.

[5] Liangyu Chen, Xiaojie Chu, Xiangyu Zhang, and Jian Sun. 2022. Simple baselines for image restoration. *arXiv preprint arXiv:2204.04676* (2022).

[6] Yuanying Dai, Dong Liu, and Feng Wu. 2017. A convolutional neural network approach for post-processing in HEVC intra coding. In *Proceedings of the International Conference on Multimedia Modeling*. 28–39.

[7] Dandan Ding, Lingyi Kong, Guangyao Chen, Zoe Liu, and Yong Fang. 2019. A switchable deep learning approach for in-loop filtering in video coding. *IEEE Transactions on Circuits and Systems for Video Technology* 30, 7 (2019), 1871–1887.

[8] Dandan Ding, Junjie Wang, Guangkun Zhen, Debargha Mukherjee, Urvang Joshi, and Zhan Ma. 2023. Neural adaptive loop filtering for video coding: Exploring multi-hypothesis sample refinement. *IEEE Transactions on Circuits and Systems for Video Technology*. Early access, March 22, 2023.

[9] Chao Dong, Yubin Deng, Chen Change Loy, and Xiaoou Tang. 2015. Compression artifacts reduction by a deep convolutional network. In *Proceedings of the IEEE International Conference on Computer Vision*. 576–584.

[10] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. 2020. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929* (2020).

[11] Chelsea Finn, Pieter Abbeel, and Sergey Levine. 2017. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the International Conference on Machine Learning*. 1126–1135.

[12] Chelsea Finn and Sergey Levine. 2017. Meta-learning and universality: Deep representations and gradient descent can approximate any learning algorithm. *arXiv preprint arXiv:1710.11622* (2017).

[13] C. Fu, E. Alshina, A. Alshin, Y. Huang, C. Chen, C. Tsai, C. Hsu, S. Lei, J. Park, and W. Han. 2012. Sample adaptive offset in the HEVC standard. *IEEE Transactions on Circuits and Systems for Video Technology* 22, 12 (2012), 1755–1764.

[14] Xueyang Fu, Zheng-Jun Zha, Feng Wu, Xinghao Ding, and John Paisley. 2019. JPEG artifacts reduction via deep convolutional sparse coding. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2501–2510.

[15] Erin Grant, Chelsea Finn, Sergey Levine, Trevor Darrell, and Thomas Griffiths. 2018. Recasting gradient-based meta-learning as hierarchical Bayes. *arXiv preprint arXiv:1801.08930* (2018).

[16] Zhenyu Guan, Qunliang Xing, Mai Xu, Ren Yang, Tie Liu, and Zulin Wang. 2019. MFQE 2.0: A new approach for multi-frame quality enhancement on compressed video. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 43, 3 (2019), 949–963.

[17] Meng-Hao Guo, Tian-Xing Xu, Jiang-Jiang Liu, Zheng-Ning Liu, Peng-Tao Jiang, Tai-Jiang Mu, Song-Hai Zhang, Ralph R. Martin, Ming-Ming Cheng, and Shi-Min Hu. 2022. Attention mechanisms in computer vision: A survey. *Computational Visual Media* 8 (2022), 331–368.

[18] Kai Han, Yunhe Wang, Hanting Chen, Xinghao Chen, Jianyuan Guo, Zhenhua Liu, Yehui Tang, An Xiao, Chunjing Xu, Yixing Xu, Zhaohui Yang, Yiman Zhang, and Dacheng Tao. 2020. A survey on visual transformer. *arXiv preprint arXiv:2012.12556* 2, 4 (2020).

[19] Jie Hu, Li Shen, and Gang Sun. 2018. Squeeze-and-excitation networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 7132–7141.

[20] Xuecai Hu, Haoyuan Mu, Xiangyu Zhang, Zilei Wang, Tieniu Tan, and Jian Sun. 2019. Meta-SR: A magnification-arbitrary network for super-resolution. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 1575–1584.

[21] Zhijie Huang, Jun Sun, Xiaopeng Guo, and Mingyu Shang. 2021. Adaptive deep reinforcement learning-based in-loop filter for VVC. *IEEE Transactions on Image Processing* 30 (2021), 5439–5451.

[22] Zhijie Huang, Jun Sun, Xiaopeng Guo, and Mingyu Shang. 2021. One-for-all: An efficient variable convolution neural network for in-loop filter of VVC. *IEEE Transactions on Circuits and Systems for Video Technology* 32, 4 (2021), 2342–2355.

[23] Andrey Ignatov, Radu Timofte, Thang Van Vu, Tung Minh Luu, Trung X. Pham, Cao Van Nguyen, Yongwoo Kim, Jae-Seok Choi, Munchurl Kim, Jie Huang, Jiewen Ran, Chen Xing, Xingguang Zhou, Pengfei Zhu, Mingrui Geng, Yawei LI, Eirikur Agustsson, Shuhang Gu, Luc Van Gool, Etienne de Stoutz, Nikolay Kobyshev, Kehui Nie, Yan Zhao, Gen Li, Tong Tong, Qinquan Gao, Liu Hanwen, Pablo Navarrete Michelini, Zhu Dan, Hu Fengshuo, Zheng Hui, Xiumei Wang, Lirui Deng, Rang Meng, Jinghui Qin, Yukai Shi, Wushao Wen, Liang Lin, Ruicheng Feng, Shixiang Wu, Chao Dong, Yu Qiao, Subeesh Vasu, Nimisha Thekke Madam, Praveen Kandula, A. N. Rajagopalan, Jie Liu, and Cheolkon Jung. 2018. PIRM challenge on perceptual image enhancement on smartphones: Report. In *Proceedings of the European Conference on Computer Vision Workshops (ECCV'18)*.

[24] Chuanmin Jia, Shiqi Wang, Xinfeng Zhang, Shanshe Wang, Jiaying Liu, Shiliang Pu, and Siwei Ma. 2019. Content-aware convolutional neural network for in-loop filtering in high efficiency video coding. *IEEE Transactions on Image Processing* 28, 7 (2019), 3343–3356.

[25] Wei Jia, Li Li, Zhu Li, Xiang Zhang, and Shan Liu. 2021. Residual-guided in-loop filter using convolution neural network. *ACM Transactions on Multimedia Computing, Communications, and Applications* 17, 4 (2021), 1–19.

[26] Marta Karczewicz, Nan Hu, Jonathan Taquet, Ching-Yeh Chen, Kiran Misra, Kenneth Andersson, Peng Yin, Taoran Lu, Edouard François, and Jie Chen. 2021. VVC in-loop filters. *IEEE Transactions on Circuits and Systems for Video Technology* 31, 10 (2021), 3907–3925.

[27] Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[28] Lingyi Kong, Dandan Ding, Fuchang Liu, Debargha Mukherjee, Urvang Joshi, and Yue Chen. 2020. Guided CNN restoration with explicitly signaled linear combination. In *Proceedings of the 2020 IEEE International Conference on Image Processing (ICIP'20)*. IEEE, Los Alamitos, CA, 3379–3383.

[29] Y. Li, K. Zhang, J. Li, L. Zhang, H. Wang, M. Coban, A. M. Kotra, M. Karczewicz, F. Galpin, K. Andersson, J. Ström, D. Liu, and R. Sjöberg. 2022. EE1-1.6: Deep in-loop filter with fixed point implementation. *WG 05 MPEG Joint Video Coding Team(s) with ITU-T SG 16, JVET-AA0111* (July 2022).

[30] Yue Li, Li Zhang, and Kai Zhang. 2023. iDAM: Iteratively trained deep in-loop filter with adaptive model selection. *ACM Transactions on Multimedia Computing, Communications, and Applications* 19, 1s (2023), Article 34, 22 pages.

[31] Kai Lin, Chuanmin Jia, Xinfeng Zhang, Shanshe Wang, Siwei Ma, and Wen Gao. 2022. NR-CNN: Nested-residual guided CNN in-loop filtering for video coding. *ACM Transactions on Multimedia Computing, Communications, and Applications* 18, 4 (2022), 1–22.

[32] Chao Liu, Heming Sun, Jiro Katto, Xiaoyang Zeng, and Yibo Fan. 2020. A learning-based low complexity in-loop filter for video coding. In *Proceedings of the 2020 IEEE International Conference on Multimedia and Expo Workshops (ICMEW'20)*. IEEE, Los Alamitos, CA, 1–6.

[33] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. 2021. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 10012–10022.

[34] Di Ma, Fan Zhang, and David R. Bull. 2020. MFRNet: A new CNN architecture for post-processing and in-loop filtering. *IEEE Journal of Selected Topics in Signal Processing* 15, 2 (2020), 378–387.

[35] Haoyu Ma, Bingchen Gong, and Yizhou Yu. 2022. Structure-aware meta-fusion for image super-resolution. *ACM Transactions on Multimedia Computing, Communications, and Applications* 18, 2 (2022), 1–25.

[36] Sachin Mehta and Mohammad Rastegari. 2022. Separable self-attention for mobile vision transformers. *arXiv preprint arXiv:2206.02680* (2022).

[37] Nicola Messina, Giuseppe Amato, Andrea Esuli, Fabrizio Falchi, Claudio Gennaro, and Stéphane Marchand-Maillet. 2021. Fine-grained visual textual alignment for cross-modal retrieval using transformer encoders. *ACM Transactions on Multimedia Computing, Communications, and Applications* 17, 4 (2021), 1–23.

[38] Inbar Mosseri, Maria Zontak, and Michal Irani. 2013. Combining the power of internal and external denoising. In *Proceedings of the IEEE International Conference on Computational Photography (ICCP'13)*. IEEE, Los Alamitos, CA, 1–9.

[39] Alex Nichol, Joshua Achiam, and John Schulman. 2018. On first-order meta-learning algorithms. *arXiv preprint arXiv:1803.02999* (2018).

[40] A. Norkin, G. Bjontegaard, A. Fuldseth, M. Narroschke, M. Ikeda, K. Andersson, M. Zhou, and G. Van der Auwera. 2012. HEVC deblocking filter. *IEEE Transactions on Circuits and Systems for Video Technology* 22, 12 (2012), 1746–1754.

[41] Zhaoqing Pan, Xiaokai Yi, Yun Zhang, Byeungwoo Jeon, and Sam Kwong. 2020. Efficient in-loop filtering based on enhanced deep convolutional neural networks for HEVC. *IEEE Transactions on Image Processing* 29 (2020), 5352–5366.

[42] Seobin Park, Jinsu Yoo, Donghyeon Cho, Jiwon Kim, and Tae Hyun Kim. 2020. Fast adaptation to super-resolution networks via meta-learning. In *Proceedings of the European Conference on Computer Vision*. 754–769.

[43] Jae Woong Soh, Sunwoo Cho, and Nam Ik Cho. 2020. Meta-transfer learning for zero-shot super-resolution. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 3516–3525.

[44] Heiko Schwarz, Thomas Schierl, and Detlev Marpe. 2014. Block structures and parallelism features in HVEC. In *High Efficiency Video Coding (HEVC): Algorithms and Architectures*, Vivienne Sze, Madhukar Budagavi, and Gary J. Sullivan (Eds.). Integrated Circuits and Systems, Vol. 39. Springer, 49–90.

[45] Chia-Yang Tsai, Ching-Yeh Chen, Tomoo Yamakage, In Suk Chong, Yu-Wen Huang, Chih-Ming Fu, Takayuki Itoh, Takashi Watanabe, Takeshi Chujoh, Marta Karczewicz, and Shaw-Min Lei. 2013. Adaptive loop filtering for video coding. *IEEE Journal of Selected Topics in Signal Processing* 7, 6 (2013), 934–945.

[46] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in Neural Information Processing Systems* 30 (2017), 1–11.

[47] Dezhao Wang, Sifeng Xia, Wenhan Yang, and Jiaying Liu. 2021. Combining progressive rethinking and collaborative learning: A deep framework for in-loop filtering. *IEEE Transactions on Image Processing* 30 (2021), 4198–4211.

[48] Jiahao Wang, Songyang Zhang, Yong Liu, Taiqiang Wu, Yujiu Yang, Xihui Liu, Kai Chen, Ping Luo, and Dahua Lin. 2023. RIFormer: Keep your vision backbone effective but removing token mixer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 14443–14452.

[49] Shen Wang, Yibing Fu, Chen Zhu, Li Song, and Wenjun Zhang. 2022. Low-complexity multi-model CNN in-loop filter for AVS3. In *Proceedings of the 2022 IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP'22)*. IEEE, Los Alamitos, CA, 1630–1634.

[50] Tengfei Wang, Hao Ouyang, and Qifeng Chen. 2021. Image inpainting with external-internal learning and monochromic bottleneck. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 5120–5129.

[51] Wenhai Wang, Enze Xie, Xiang Li, Deng-Ping Fan, Kaitao Song, Ding Liang, Tong Lu, Ping Luo, and Ling Shao. 2021. Pyramid Vision Transformer: A versatile backbone for dense prediction without convolutions. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 568–578.

[52] Xintao Wang, Liangbin Xie, Chao Dong, and Ying Shan. 2021. Real-ESRGAN: Training real-world blind super-resolution with pure synthetic data. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 1905–1914.

[53] Zhendong Wang, Xiaodong Cun, Jianmin Bao, Wengang Zhou, Jianzhuang Liu, and Houqiang Li. 2022. Uformer: A general U-shaped transformer for image restoration. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 17683–17693.

[54] Zhao Wang, Changyue Ma, Ru-Ling Liao, and Yan Ye. 2021. Multi-density convolutional neural network for in-loop filter in video coding. In *Proceedings of the 2021 Data Compression Conference (DCC'21)*. IEEE, Los Alamitos, CA, 23–32.

[55] Sanghyun Woo, Jongchan Park, Joon-Young Lee, and In So Kweon. 2018. CBAM: Convolutional block attention module. In *Proceedings of the European Conference on Computer Vision (ECCV'18)*. 3–19.

[56] Jie Xiao, Xueyang Fu, Feng Wu, and Zheng-Jun Zha. 2022. Stochastic window transformer for image restoration. In *Proceedings of the 36th Conference on Neural Information Processing Systems (NeurIPS'22)*.

[57] Qunliang Xing, Mai Xu, Tianyi Li, and Zhenyu Guan. 2020. Early exit or not: Resource-efficient blind quality enhancement for compressed images. In *Proceedings of the European Conference on Computer Vision*. 275–292.

[58] Yazhou Xing, Zian Qian, and Qifeng Chen. 2021. Invertible image signal processing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 6287–6296.

[59] Fuzhi Yang, Huan Yang, Jianlong Fu, Hongtao Lu, and Baining Guo. 2020. Learning texture transformer network for image super-resolution. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 5791–5800.

[60] Weihao Yu, Mi Luo, Pan Zhou, Chenyang Si, Yichen Zhou, Xinchao Wang, Jiashi Feng, and Shuicheng Yan. 2022. MetaFormer is actually what you need for vision. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 10819–10829.

[61] Syed Waqas Zamir, Aditya Arora, Salman Khan, Munawar Hayat, Fahad Shahbaz Khan, and Ming-Hsuan Yang. 2022. Restormer: Efficient transformer for high-resolution image restoration. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 5728–5739.

Received 27 October 2022; revised 24 May 2023; accepted 28 July 2023