Joint Pixel and Frequency Feature Learning and Fusion via Channel-wise Transformer for High-Efficiency Learned In-Loop Filter in VVC

Birendra Kathariya, *Student Member, IEEE,* Zhu Li, *Senior Member, IEEE,*
Geert Van der Auwera, *Senior member, IEEE*

***Abstract*—Block-based video codecs such as Versatile Video Coding (VVC)/H.266, High Efficiency Video Coding (HEVC)/H.265, Advanced Video Coding (AVC)/H.264 etc. inherently introduces compression artifacts. Although these codecs have in-loop filters to correct these distortions, they are not always effective due to the complexity of the noise. Recently, deep-learning approaches emerged as a promising solution for in-loop filtering. However, most of the previous approaches were designed solely for learning from images and neglected the high-frequency signals present in the reconstructed video frames. Furthermore, some previous methods employed a multi-level feature-extraction and feature-fusion strategy to enhance performance. However, they utilized complex feature-extractors while relying on naive feature-fusion methods. In this article, we propose a novel framework called *TSF-Net*, which jointly learns from both the pixel (spatial) and frequency-decomposed information and through powerful capability of a channel-wise transformer, it fuses both these information to improve performance. Our approach deviates from previous approaches by employing a simple feature-extractor coupled with an advanced transformer-based feature-fusion module. Simultaneously, *TSF-Net* introduces a few fundamental modifications in the multi-head self-attention module of the channel-wise transformer to make it computationally efficient. Our experimental results show that the proposed *TSF-Net* achieves a Bjøntegaard Delta (BD) - bitrate saving of up to 10.258% for the luma (Y) component under all-intra (AI) profile outperforming the VVC baseline and other state-of-the-art methods. Moreover, the proposed *TSF-Net* with an efficient channel-wise transformer is twice as efficient as *TSF-Net* with a vanilla channel-wise transformer.**

***Index Terms*—in-loop filters, versatile video coding, convolutional neural network, channel-wise transformer, feature fusion**

I. Introduction

ADVANCEMENTS in the internet, computing, and display technologies have revolutionized the way we consume video content. Today, the majority of internet traffic is dominated by video data from various sources such as streaming, conferencing, surveillance, and more. Video compression technology is the cornerstone of these video-processing applications, enabling the efficient transmission and storage of large amounts of video data. Over the years, with the continuous improvement of video capture and processing technology, several video compression standards have been developed to meet the increasing demand. These standards include Advanced Video Coding (AVC)/H.264 [1], High Efficiency Video Coding (HEVC)/H.265 [2], and Versatile Video Coding (VVC)/H.266 [3] among others.

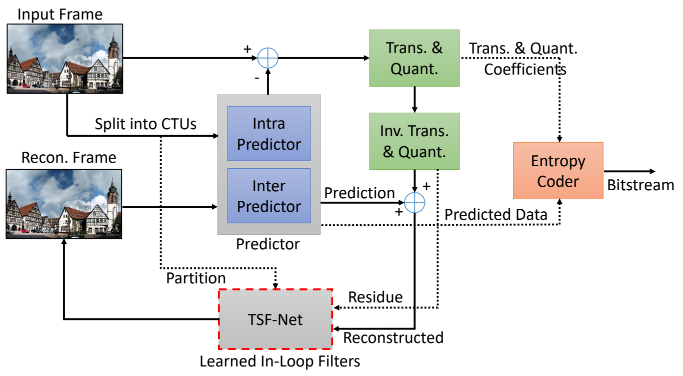

Fig. 1. Proposed *TSF-Net* as an In-loop Filter in VVC architecture.

Currently, Versatile Video Coding (VVC)/H.266 [3] has been established as the next generation video coding standard by the Joint Video Experts Team (JVET), with over 30% performance improvement over its predecessor, High Efficiency Video Coding (HEVC)/H.265. VVC was developed to meet the quality-of-experience (QoE) demands of end-users as the resolution of displays (and videos) increased, supporting ultra-high definition (UHD) videos up to 16K. Secondly, VVC is designed to be versatile, providing coding and transport support for a wide range of applications and content types, such as conventional video streaming, screen content optimization, 360° video for AR/VR, live broadcasting, and ultra-low latency applications.

Nonetheless, VVC is not completely a new technology but rather an evolution of HEVC, wherein most of the technologies from HEVC are further improved, and many new coding tools are invented to account for larger resolutions and different contents. This implies that VVC is still a block-based hybrid video coding standard, where a picture is partitioned into smaller, non-overlapping blocks that go through operations like prediction, transform, quantization, etc. in an independent

Birendra Kathariya and Zhu Li are with the Department of Computer Science and Electrical Engineering, University of Missouri-Kansas City, MO 64110 USA (email: bkkvh8@umsystem.edu; zhu.li@ieee.org).
Geert Van der Auwera is with Qualcomm Technologies Inc., San Diego, CA 92121 USA (e-mail: geertv@qti.qualcomm.com).
Manuscript received March 23, 2023

manner. These blocks are named based on the operations applied, such as *prediction unit* (PU), *coding unit* (CU), *transform unit* (TU), etc.

Like HEVC, VVC also suffers from inherent compression artifacts such as blocking, ringing, blurring, and mosquito noise due to block-based operations. In intra-frames (I-frames), the transform coefficients from neighboring blocks typically undergo quantization differently. Similarly, in prediction-frames (P-frames) and bidirectional prediction-frames (B-frames), the neighboring blocks are likely to use different motion vectors, resulting in discontinuities at the block boundary. Simultaneously, the quantization applied to transform coefficients and prediction residues at the block level results in ringing, blurring, and other noise. If these compression artifacts are not handled properly, they can degrade not only the subjective and objective quality of the current frame but also propagate to successive frames if referenced by the P- or B-frame.

VVC, similar to HEVC, is equipped with in-loop filters [4] that correct compression artifacts. The in-loop filters in VVC include three types of traditional filters, namely deblocking filter (DBF) [5], sample adaptive offset (SAO) [6], and adaptive loop filter (ALF) [7]. These filters are applied to a reconstructed picture one after another in the given order. DBF is responsible for suppressing the blocking artifacts at block boundaries, while SAO and ALF filters are designed to remove artifacts caused by quantization. Nonetheless, these filters' quality restoration capability is sub-optimal and leaves a large room for improvement.

In recent times, deep learning has achieved numerous breakthroughs in the field of computer vision. Deep learning methods, such as convolutional neural networks (CNNs) [8] and vision transformers [9], have already made great strides in applications such as image super-resolution [10]–[12] and image restoration tasks such as denoising, deblurring, dehazing, and image enhancement [13]–[20]. CNN-extracted features are locally correlated, whereas transformers can extract features with long-range correlation. Recently, hybrid architectures that employ both CNN and transformers [15], [21] have emerged. These hybrid models are capable of extracting deep features with better super-resolving and image-restoration power.

To improve the feature representation power of CNN, researchers are now proposing complex architectures [16]–[18] that operate at multi-scale feature-size. With this "coarse-to-fine" strategy, features of various spatial sizes are extracted (multi-scale), which are then progressively fused from coarser to finer scales until the original scale is achieved. Other approaches inspired by Inception-like-network [22] utilize kernels of variable sizes to derive features with various receptive-field and fuse them to enrich the feature semantics [23]. Prior works [24]–[29] have also demonstrated the benefit of utilizing frequency decomposition (Discrete Cosine Transform (DCT) and Wavelet Transform) in conjunction with CNN to improve feature representation for learning low-level vision tasks. DCT and Wavelet-transform decompose a signal (image) into multiple frequency bands that can be utilized as crucial prior information for image restoration tasks where high-frequency contents such as noises need to be discriminated from the signal.

Inspired by the success of CNN in image restoration tasks, researchers have successfully implemented CNN to fill the performance gap of the in-loop filter of HEVC and VVC. Plenty of works have also implemented CNN as a post-processing block in both HEVC and VVC to improve the final picture quality. A few works have even utilized CNN to improve or replace conventional coding tools like intra-prediction [30], inter-prediction [31], etc. Researchers have also trained CNN-based auto-encoder-style neural networks in an end-to-end fashion for applications like image compression [32] and video compression [33]. In such approaches, images and video frames are represented by features at the bottleneck layer of an auto-encoder. Most of the learning-based in-loop filter works are largely inspired by CNN-based image restoration works.

In this article, we design a hybrid model consisting of both convolutional and transformer layers for feature extraction and feature fusion, respectively, as an in-loop filter for VVC. We refer it as *Transformer-based Spatial and Frequency-Decomposed Feature Fusion Network (TSF-Net)*. Moreover, we not only use pixel information but also frequency-decomposed (DCT'ed image) information to improve the feature representation capability of the model, which was largely ignored by prior articles while designing a learned in-loop filter. We treat each DCT coefficient as a separate channel, thereby creating DC and AC channels. Convolutional layers are then utilized to extract deep features from pixel and DCT'ed information, and a transformer layer is utilized to fuse DCT features into pixel features.

The transformer utilized in this work has a self-attention module designed to perform attention along the channel where each channel is treated as a token. We prepare pixel input by pixel-unshuffling an image, which distributes local pixels along the channel. Similarly, DCT'ed input is prepared by separating DCT coefficients along the channel. This requires a proper mechanism to extract features from local pixels as well as DC and AC coefficients distributed along the channel. Thus, we consider a transformer layer with similar functionality called *Spectral-wise Multi-head Self-attention (S-MSA)* from the MST++ [34] article. Nonetheless, we fundamentally redesign the S-MSA layer to accomplish feature fusion between pixel (spatial) and DCT'ed (frequency-decomposed) features efficiently.

The contributions of this article are summarized below.

- We propose *TSF-Net*, a learned in-loop filter for VVC, which learns from both spatial (pixel) and frequency-decomposed information and offers state-of-the-art performance under all-intra (AI) profile for luma (Y) component.
- Our proposed *TSF-Net* is highly scalable as it can be constructed simply by cascading multiple *Residual Fusion Block (RFB)* similar to *Residual Blocks* in *ResNet* architectures. Each *RFB* bundles together convolution-based feature processing and transformer-based feature fusion operation.
- We utilize *S-MSA* from *MST++*, a channel-wise transformer layer, and redesign it for feature fusion. We

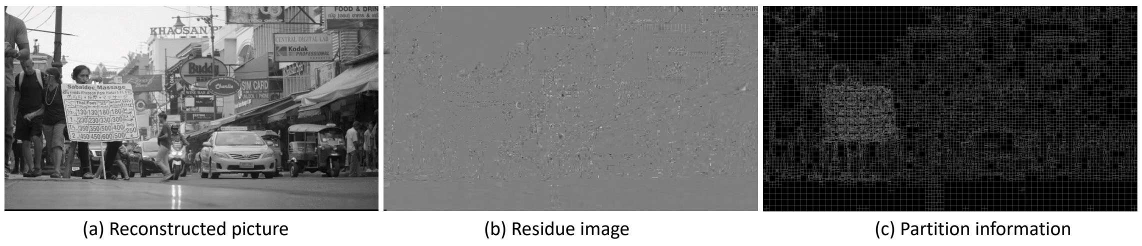

Fig. 2. Luma and its side-information from VVC utilized as input to the *TSF-Net*.

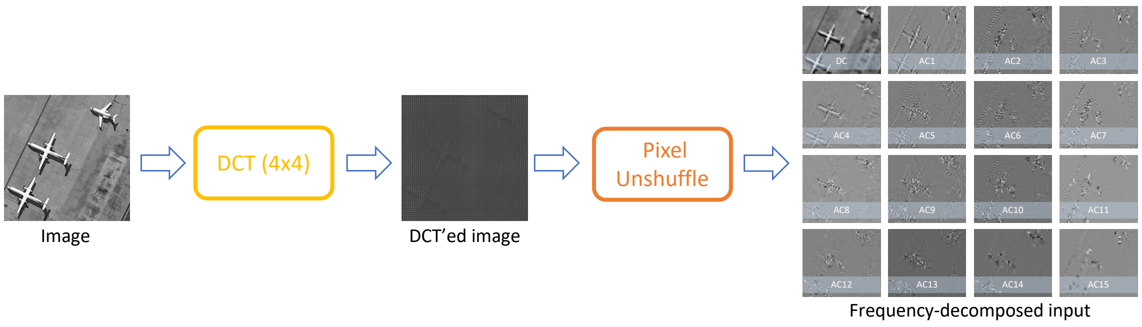

Fig. 3. Frequency-decomposed input generation pipeline. First 4-point 2D-DCT is applied on a $N \times N$ image and then it is pixel-unshuffled on 4x4 block to get 16-channel $(N/4) \times (N/4)$ frequency-decomposed image.

further propose efficient *S-MSA* (*E-SMSA*) by making fundamental modifications in the self-attention module. *E-SMSA* improves the feature processing time of *TSF-Net* nearly by twice.

- We utilize large patch inputs and while pixel-unshuffle them, we are able to decrease the feature processing time and increase the receptive field. Additionally, *RFB* processes spatial and frequency-decomposed features in parallel, contributing to a further reduction in feature processing time.

The rest of the article is organized as follows. In Section II, we go over the prior in-loop filtering technologies related to classical and deep-learning approaches. In Section III, we lay out the details of our proposed architecture and related methods. Section IV describes the implementation and training details of our method as an in-loop filter. In Section V, we present the findings of the experiments processes. Finally, we conclude our work and its findings in Section VI.

II. Related Work

A. Traditional In-loop Filters

HEVC adopts two filters: deblocking filter (DBF) [5] and Sample Adaptive Offset (SAO) [6], as in-loop filters. Deblocking filter is first applied to the reconstructed sample which attenuates the discontinuities at the block boundary. The deblocked picture is then processed by the SAO to further mitigate the ringing artifacts and corrects the signal (image) by applying the offsets to the pixels. The offset values which are calculated based on pixels' statistics are sent to the decoder. VVC keeps both DBF and SAO filters, however, adds three more filters: Luma Mapping Chroma Scaling (LMCS) [35], Adaptive Loop Filter (ALF) [7] and, Cross-Component Adaptive Loop Filter (CC-ALF) [36]. LMCS is implemented to improve coding efficiency through two processes: luma mapping (LM) and chroma scaling (CS). LM remaps the luma code values within the complete codeword range at a specific bit-depth. CS compensates for the impact of luma mapping on the relative chroma coding bit costs. ALF is designed to minimize the mean square error (MSE) between the original and reconstructed picture. The idea of ALF is to classify non-overlapping blocks based on their local sample gradient and apply a specific Wiener-based filter among many to improve signal fidelity. The type of filter used is signaled in the bitstream. CC-ALF performs the Wiener-based filtering correction on the chroma sample utilizing the co-located and corrected luma sample. These filters are primarily hand-crafted and statistical-based. While ALF has a slight learning capability, its efficiency is very much limited. This leaves a large room for improvement to exploit.

B. Learned In-loop Filters

A plethora of articles already exist that successfully implements convolution neural networks (CNN) as both in-loop filters and post-processing block in HEVC and VVC. Recently, the powerful capability of vision transformers (ViT) is also being exploited in the image and video compression domain. In this section, we will discuss some notable previous learning-based approaches for HEVC and VVC.

1) Learning-based approaches for HEVC: One of the early works in learned in-loop filters is IFCNN [37]. IFCNN is an SRCNN [10] (an image super-resolution) based architecture implemented as an in-loop filter in HEVC by replacing SAO. This 3-layer CNN architecture outperformed HEVC by up to 2.8% and 2.6% in low-delay(LD) and random-access(RA) configurations. Similarly, [38] proposed MIF-Net, a multi-frame in-loop filter for HEVC by replacing both DBF and SAO. MIF-Net includes a reference frame selector (RFS) that selects the reference frames with the best quality and content similarity for the current unfiltered frame. Then, MIF-Net utilizes both spatial and temporal information across multiple frames to enhance the current unfiltered frames. With this strategy, MIF-Net was able to outperform the HEVC baseline and other state-of-arts approaches. Likewise [39] proposed Squeeze-and-Excitation CNN (SEFCNN), a switchable in-loop filter positioned in parallel to HEVC's in-loop filter. By implementing two subnets: Feature Extraction (FEX) and Feature Enhancement (FEN), SEFCNN was able to learn very high-quality features. FEX utilized convolutional layers to extract deeper features, while FEN employed squeeze-and-excitation layers to enhance features by enabling channel interaction. SEFCNN achieved average of 9.96%, 8.04%, and 7.60% BD-Rate saving on AI, LDP and RA configurations, respectively, compared to HEVC. Further, [40] proposed a Recursive Residual Convolutional Network (RRCNN) in-loop filter for HEVC. In RRCNN, the authors demonstrated the increased capability of CNN by proposing multi-path residual and recursive learning. With recursive learning, RRCNN utilized the same layers repeatedly for deeper feature extraction. Unlike previous works, this work trained the same model for multiple bitrates achieving an average bitrate savings of up to 8.7% on intra-frames.

Inspired by RRCNN, [41] proposed Lightweight Multiattention Recursive Residual CNN-based In-loop Filter (LMA-RRCNN) as an alternate to HEVC's in-loop filter. LMA-RRCNN, similar to RRCNN, utilized parameter sharing and feature reuse to reduce model parameters and increase model depth, while simultaneously learning features at multiple spatial scales and frequently fusing them. LMA-RRCNN was also able to handle video compressed at various bitrates and of various frame-types via single model while achieving excellent bitrate savings upto 13.7% and 11.87% in AI and RA profiles, respectively. Another work [42], proposed Frame-wise filtering for Quality Enhancement based on CNN (FQE-CNN), which also adopted the multi-level feature extraction and feature fusion approach similar to LMA-RRCNN. FQE-CNN utilized an inception-like residual learning block (IResLB) to extract features and occasionally aggregated the features between spatial levels through concatenation and convolution. With this multi-level feature learning strategy, FQE-CNN achieved superior performance compared to the HEVC baseline, surpassing it by 11.1% in all-intra (AI) configurations.

2) Learning-based approaches for VVC: With the emergence of VVC as the latest video coding framework, a plethora of fascinating new research has been generated on the topic of learned in-loop filters. For example, [43] proposed a dense residual convolutional neural network (DRN) based in-loop filter (DRNLF) for VVC and placed it after DFB and before SAO and ALF. By utilizing dense shortcuts in DenseNet [44] and feature reuse, DRNLF achieved 1.52%, 1.45%, and 1.54% BD-rate saving in AI, RA, and LD coding configurations. Similarly, [45] proposed a multi-gradient convolutional neural network-based in-loop filter (MGNLF) for VVC, where the network also considers divergence and second-derivative of the frame along with the frame itself to improve the quality of the frame. By utilizing contour and structural information of a frame, MGNLF was able to reduce BD-rate savings up to 3.29% on average. MFRNet [46], however, is proposed as both a post-processing and an in-loop filter on both HEVC and VVC. As an in-loop filter, MFRNet was placed after SAO (for HEVC) and ALF (for VVC). MFRNet utilized highly dense shortcuts for multi-level feature reuse and achieved coding gain (BD-rate VMAF) of up to 16% and 5.1% for HEVC and VVC, respectively. Similarly, [47] proposed Variable CNN (VCNN), an in-loop filter specifically designed to handle videos compressed at different quantization parameters (QPs) and frame-types (FT) using a single model. VCNN incorporated a QP attention module (QPAM) and a FT attention module (FTAM) in its residual block (RB). These modules recalibrated the features through channel-attention, allowing VCNN to adapt to a wide range of QPs and FTs. VCNN outperformed VVC baseline by upto 3.63%, 4.36%, 4.23%, 3.56% in AI, LDP, LD and RA configurations. Likewise, [48] proposed MSCNN, in which two U-Net like architectures are utilized to extract features from reference and current frames while aggregating both feature through gated-fusion. MSCNN achieved 3.762% bitrate saving in AI configuration. Likewise, as a response to the call for proposal (CfP) on "Neural networks on video coding" [49], JVET received [50] as a contribution from ByteDance. This work proposed Deep In-loop filter with Adaptive Model selection (DAM) and is currently adopted as part of the standard for further exploration.

Interestingly, all previous works ignored high-frequency information while designing an in-loop filter. Nonetheless, [50] utilized both spatial(pixel) and frequency decomposed (DCT) information and designed a CNN architecture that extracted both pixel and DCT features while fusing them at multiple stages. The proposed work, on average, achieved a 9.7% BD-rate reduction compared to the VVC baseline. In this work, convolution-based feature fusion between pixel and DCT features was proposed. Later, the authors extended the work by proposing MSTFNet [51] where convolution-based feature fusion was replaced by transformer-based feature fusion and significantly improved the BD-rate savings. MSTFNet utilized a transformer with channel-wise self-attention to aggregate pixel and DCT features at multiple stages. This work demonstrated a successful implementation of a transformer along with CNN as an in-loop filter.

To improve the performance of learned in-loop filters, researchers extended single-level architecture to a more advanced multi-level architecture, incorporating feature interaction among these levels. However, previous methods [42] [41] primarily focused more on designing a complex feature-extractor at each levels but relied on naive feature-fusion approaches. In this article, we take a different approach by utilizing a simple feature-extractor while implementing transformers-based a more advanced feature-fusion module. Furthermore, our method involves fusion of spatial and frequency-decomposed features. This is in contrast to previous approaches where feature-fusion occurs only between spatial-

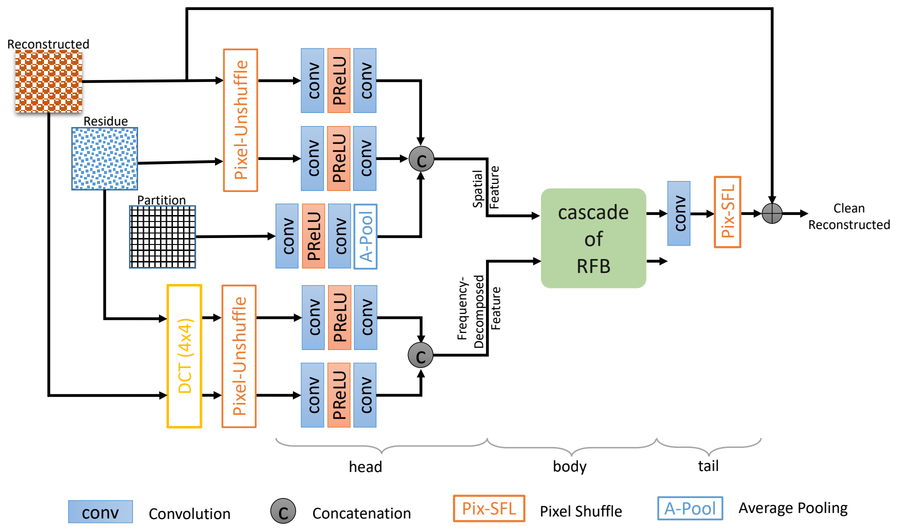

Fig. 4. Overall architecture of our proposed *Transformer-based Spatial and Frequency-decomposed Feature Fusion Network (TSF-Net)*. *TSF-Net* is divided into 3 parts: head, body and tail. Reconstructed picture, residue and partition information from VVC is processed to generate five different types of inputs: 3 spatial inputs and 2 frequency-decomposed inputs. The Head of *TSF-Net* transforms spatial and frequency-decomposed inputs into two types of features: spatial and frequency-decomposed features. The body includes a series of *Residual Fusion Blocks (RFB)* which take both types of features and fuse them into the spatial features. The spatial feature output from the body is finally processed by the tail to produce a clean reconstructed picture.

features but at different scales. Our current work (*TSF-Net*) can be considered an extension of our prior work MSTFNet [51]. Although both employ spatial and frequency-decomposed inputs and a transformer-based feature aggregation scheme (*S-MSA*), we drastically reduce the number of trainable parameters in *TSF-Net* by revamping the overall architecture completely. In contrast to MSTFNet, which only fuses features at 3 stages, *TSF-Net* fuses feature at 20 stages using 20 residual fusion blocks (RFB). Furthermore, *TSF-Net* utilizes efficient *S-MSA* that is twice as efficient as the vanilla *S-MSA* used in MSTFNet.

III. Proposed Method

In the decoder of VVC architecture, we integrate our learned in-loop filter, Transformer-based Spatial and Frequency-Decomposed Feature Fusion Network (TSF-Net), by replacing all the modules: DBF, SAO, ALF (CCALF) of the in-built in-loop filter. This integration is illustrated in figure 1. Unlike the in-built in-loop filter of VVC, *TSF-Net* does not utilize any filter control data and hence, it is not communicated to the entropy encoder. In this work, we consider three types of information: reconstructed picture, partition information, and residual image as input to the *TSF-Net*. The reconstructed picture serves as the primary input while partition and residual image are crucial side information that convey information related to the block boundary and prediction error respectively. The proposed *TSF-Net* is presented in figure 4. It receives five distinct types of inputs: three spatial inputs and two frequency-decomposed inputs, generated from the reconstructed picture, partition, and residual image. In this section, we will first describe the methods we adopted to prepare the inputs for *TSF-Net*, followed by the details of *TSF-Net* itself.

A. Generation of Spatial and Frequency-Decomposed Inputs

The proposed *TSF-Net* utilizes three types of information: reconstructed (I), partition (P), and residual (R) image as inputs. An example of these input images are shown in figure 2. The reconstructed picture (I) is an unfiltered picture generated after addition of prediction image and residual image. The residual image (R) is the difference image of the original and the predicted image. The partition (P) image is the coding-unit (CU) level block partition information. In this study, we apply *TSF-Net* only on the luma (Y) component. Therefore, I represents the luma component of the reconstructed picture, whereas R and P correspond to the associated residue and partition information respectively. Let $2H$ and $2W$ be the height and width of these I, R, and P images.

In the next step, three spatial inputs are created from I, R, and P images, and two frequency-decomposed inputs from I and R images, respectively. The I and R images are pixel-unshuffled on a 2×2 block to create two spatial inputs, $I_s \in \mathbb{R}^{4\times H\times W}$ and $R_s \in \mathbb{R}^{4\times H\times W}$, respectively. Here, pixel-unshuffling on a 2×2 block converts an $2H \times 2W$ image to a 4-channel $H \times W$ image. Since the boundary information is just a single pixel wide, the pixel-unshuffling operation would break the structure in the partition image P. Therefore, the partition image P is fed to the network unchanged, and thus spatial input $P_s = P \in \mathbb{R}^{1\times 2H\times 2W}$. Similarly, two frequency-decomposed inputs are generated by applying a 4-point 2D discrete cosine transform (DCT) on the reconstructed I and residual R images. The resulting image has a DC coefficient and 15 AC coefficients in every non-overlapping 4×4 block. In the next step, pixel-unshuffling is applied on the non-overlapping 4×4 blocks of the $2H \times 2W$ sized DCT'ed image to obtain a $16 \times (H/2) \times (W/2)$ image, where the first channel is a DC-image and rest of the 15 channels are the AC-images. Let $I_f \in \mathbb{R}^{16\times(H/2)\times(W/2)}$, $R_f \in \mathbb{R}^{16\times(H/2)\times(W/2)}$

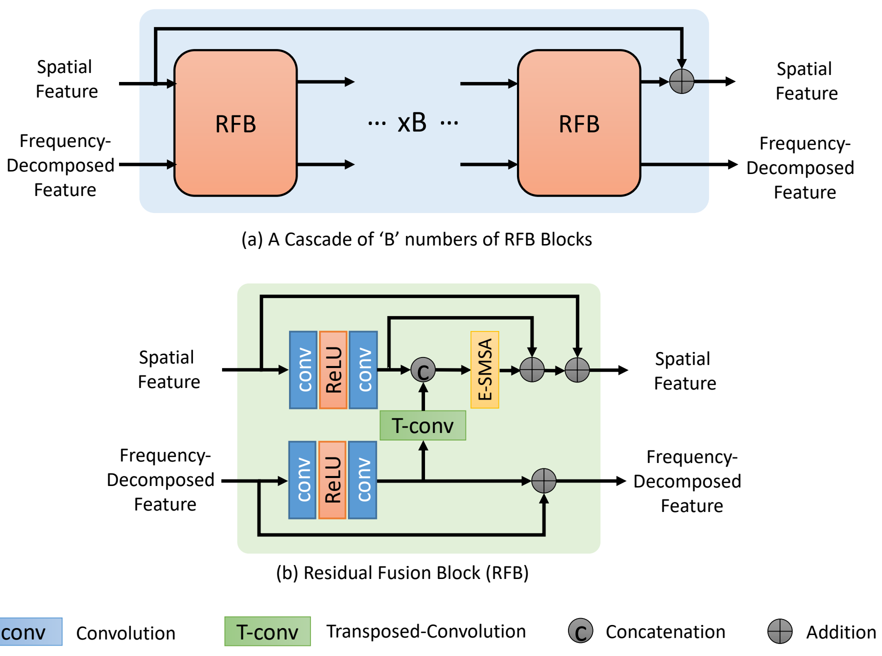
Spatial Feature
Frequency-Decomposed Feature
RFB
··· xB ···
RFB
Spatial Feature
Frequency-Decomposed Feature
(a) A Cascade of 'B' numbers of RFB Blocks
Spatial Feature
conv
ReLU
conv
C
E-SMSA
T-conv
Spatial Feature
Frequency-Decomposed Feature
conv
ReLU
conv
Frequency-Decomposed Feature
(b) Residual Fusion Block (RFB)
conv Convolution
T-conv Transposed-Convolution
C Concatenation
Addition

Fig. 5. (a) Body of *TSF-Net* constitutes a cascade of *Residual Fusion Block (RFB)* with a feature-level skip connection. (b) The internal details of *RFB*. It includes two branches with a set of **"conv→relu→conv"** block for extracting deeper features from spatial and frequency-decomposed features, a cross-connection through *transposed convolution (T-conv)* for upscaling frequency-decomposed feature and an *E-SMSA* transformer layer for efficient feature fusion of concatenated spatial and frequency-decomposed features. Both branches include a skip connection making the fusion block essentially a residual block.

be the pixel-unshuffled version of the DCT'ed image of I and R, respectively. We refer to them as frequency-decomposed inputs. The steps to obtain frequency-decomposed inputs are shown in figure 3.

The spatial inputs I_s and R_s are normalized by dividing them with the peak value of $2^b - 1$, where b is the number of bits used to represent I. P_s is a binary image with the value of 1 at the boundary and 0 everywhere else, so it does not require normalization. In the DCT domain, however, the DC coefficient has the largest value and AC values gradually decrease. Thus, frequency-decomposed inputs I_f and R_f are normalized channel-wise using *"min-max"* normalization method. The channel-wise minimum and maximum values $\{min_I, max_I\} \in \mathbb{R}^{16\times1\times1}$ and $\{min_R, max_R\} \in \mathbb{R}^{16\times1\times1}$ of I_f and R_f respectively, are first computed from the training dataset. For this, we utilize 25000 randomly cropped co-located patches of size 256×256 from I and R images. The same channel-wise *"min-max"* values obtained from the training dataset are used for normalizing the frequency-decomposed inputs during inference.

B. Description of TSF-Net architecture

The overall architecture of *TSF-Net* is presented in figure 4. The proposed network is slightly inspired by EDSR [52] as it includes both global and feature-level skip connections. *TSF-Net* can be better explained by dividing it into three parts: head, body, and tail.

1) Head: The head of *TSF-Net* comprises four **"conv→PReLU→conv"** blocks and one **"conv→PReLU→conv→avg-pool"** block. The two spatial inputs I_s and R_s and two frequency-decomposed inputs I_f and R_f are processed through their respective **"conv→PReLU→conv"** blocks. We use the **"conv→PReLU→conv→avg-pool"** block to process the partition input P_s. Since P_s is at its original size $2H \times 2W$, the average-pooling layer (**"avg-pool"**) of **"conv→PReLU→conv→avg-pool"** reduces it to $H \times W$. The **"conv→PReLU→conv"** and **"conv→PReLU→conv→avg-pool"** blocks transform spatial and frequency-decomposed inputs into the initial feature representation. Let $C_n, n = \{1,2,3,4,5\}$ represent the channel-size of the initial features. Similarly, let $F_s^I \in \mathbb{R}^{C_1\times H\times W}$, $F_s^R \in \mathbb{R}^{C_2\times H\times W}$ and $F_s^P \in \mathbb{R}^{C_3\times H\times W}$ be the initial feature representation of spatial inputs I_s, R_s, P_s respectively, and $F_f^I \in \mathbb{R}^{C_4\times(H/2)\times(W/2)}$ and $F_f^R \in \mathbb{R}^{C_5\times(H/2)\times(W/2)}$ be the initial feature representation of frequency-decomposed inputs I_f, R_f, respectively. Next, spatial feature (F_s) and frequency-decomposed feature (F_f) are obtained by concatenating corresponding spatial and frequency-decomposed initial features.

$$F_s = concat(\{F_s^I, F_s^R, F_s^P\}), \quad F_f = concat(\{F_f^I, F_f^R\}) \tag{1}$$

where $concat(\cdot)$ is a concatenation operation. In this article, we configure **"conv→PReLU→conv"** and **"conv→PReLU→conv→avg-pool"** block such that initial features channel-size become $C_1 = 48, C_2 = 24, C_3 = 8, C_4 = 16, C_5 = 16$. Therefore, after concatenation $F_s \in \mathbb{R}^{C_s\times H\times W}$, $C_s = 96$ and $F_f \in \mathbb{R}^{C_f\times(H/2)\times(W/2)}$, $C_f = 32$.

2) Body: The *body* of *TSF-Net* is composed of cascading B *Residual Feature Block (RFB)*, as illustrated in figure 5 (a). An

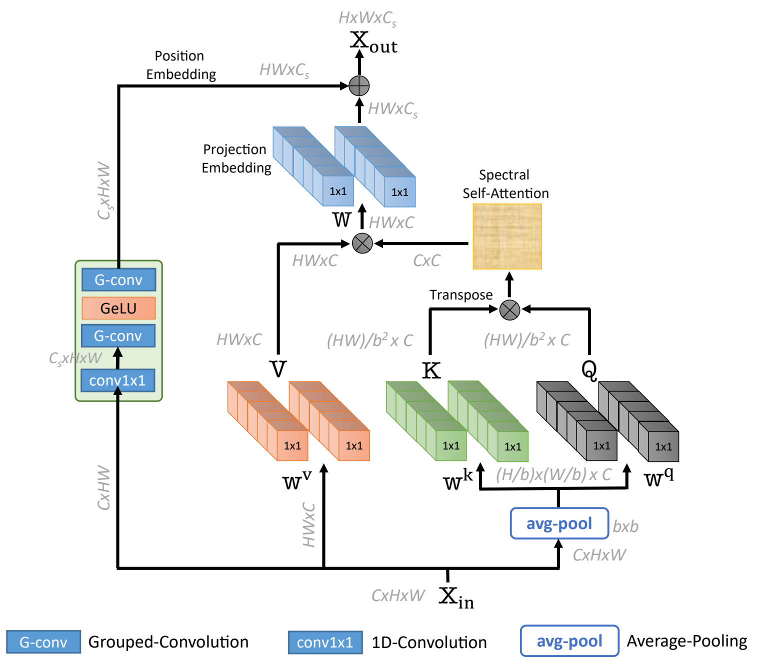

Fig. 6. Efficient Spectral-wise Multihead Self-Attention (E-SMSA).

RFB receives spatial features (F_s) and frequency-decomposed features (F_f) through its two separate inputs and produces more refined spatial features (F_s) and frequency-decomposed features (F_f) through its two outputs. In an *RFB* block, the frequency-decomposed features (F_f) get fused into the spatial features (F_s) through an *S-MSA* transformer layer. Thus, by stacking B *RFB* blocks, we fuse and refine spatial feature (F_s) and frequency-decomposed feature (F_f) at multiple stages. The *body* also includes a feature-level skip connection between the initial spatial-feature input and the final spatial-feature output of the *body*, which is illustrated in figure 5 (a). The details of the *Residual Feature Block (RFB)* are described in Section III-C.

3) Tail: The *tail* of *TSF-Net* includes a convolution layer (**"conv"**) and a pixel-shuffle layer (**"Pix-SFL"**). The final spatial feature $F_s \in \mathbb{R}^{C_s \times H \times W}$, $C_s = 96$ output from the *body* of *TSF-Net* is processed by the **"conv"** layer, which reduces the channel size to just 4 and produces $F_s \in \mathbb{R}^{C_s \times H \times W}$, where $C_s = 4$. Lastly, the **"Pix-SFL"** layer assembles the $F_s \in \mathbb{R}^{4 \times H \times W}$ into a single channel feature $F_s \in \mathbb{R}^{1 \times (2H) \times (2W)}$. The global skip-connection at the end adds the input reconstructed picture (I) back to the final spatial feature F_s, resulting in the clean reconstructed picture (I_C).

C. Residual Fusion Block (RFB) as Feature Fusion Block

The primary building block of *TSF-Net* is the *Residual Fusion Block (RFB)* which is illustrated in figure 5 (b). The inputs are the spatial (F_s) and frequency-decomposed (F_f) features, and the outputs are the deeper and more refined spatial (F_s) and frequency-decomposed (F_f) features. First, the residual features are extracted from F_s and F_f by employing two sets of **"conv→ReLU→conv"** blocks and later they are concatenated. Let, $F_s^r \in \mathbb{R}^{C_s \times H \times W}$ and $F_f^r \in \mathbb{R}^{C_f \times (H/2) \times (W/2)}$ be the residual features corresponding to F_s and F_f. Since the feature size of F_s^r is twice the size of F_f^r, the frequency-decomposed residual-feature F_f^r is upscaled by a factor of two using transposed convolution (**"T-conv"**) before concatenating it with F_s^r. The concatenated feature $F = concat(\{F_s^r, F_f^r\})$, $F \in \mathbb{R}^{C \times H \times W}$, $C = C_s + C_f$ is then passed to Efficient Spectral-wise Multi-head Self-Attention (*E-SMSA*) layer for feature fusion. The *E-SMSA* transformer layer fuses the residual features F_s^r and F_f^r into a next stage spatial residual feature $F_s^{r1} \in \mathbb{R}^{C_s \times H \times W}$. Lastly, two skip connections are introduced to the residual features F_s^{r1} by adding F_s^r and F_s. Similarly, one skip connection is also introduced to F_f^r by adding frequency-decomposed features F_f to it.

$$F_s = F_s \oplus (F_s^r \oplus F_s^{r1}), \quad F_f = F_f \oplus F_f^r \tag{2}$$

Thus, the final output of the *RFB* is again the refined spatial feature F_s and frequency-decomposed feature F_f. The skip connections from input to output convert the *fusion block* into a *residual fusion block*. The residual connection, which is inspired by the *ResNet* [53], allows us to stack multiple *RFB* blocks and make the network much deeper while still being able to converge during training. The simplicity of the design choice of *RFB* enables *TSF-Net* to scale it to various other low-level vision tasks in two ways: first, by adjusting the arbitrary number of *RFB* blocks, and second, by varying the channel size (C_s and C_f).

D. Efficient Spectral-wise Multi-Head Self-Attention (E-SMSA)

We incorporate *spectral-wise multi-head self-attention (S-MSA)* transformer layer, from MST++ [34] into our proposed network. However, we have redesigned it to serve as a channel-wise feature fusion layer within the *RFB* block. Additionally, we have modified the self-attention mechanism of the original *S-MSA* to make it computationally efficient. Since *E-SMSA* is present in each *RFB* block, these reformed aspects significantly improve computational efficiency. Figure 6 provides an illustration of *E-SMSA*.

Suppose, X_{in} is the input to the *E-SMSA* where $X_{in} = F \in \mathbb{R}^{C\times H\times W}$. X_{in} is then averaged-pooled (**"avg-pool"**) over $b \times b$ block to get $X_1 \in \mathbb{R}^{C\times(H/b)\times(W/b)}$. X_1 is then reshaped as $X_1 \in \mathbb{R}^{(HW/b^2)\times C}$ and linearly projected into query $Q \in \mathbb{R}^{(HW/b^2)\times C}$ and key $K \in \mathbb{R}^{(HW/b^2)\times C}$. Similarly, X_{in} is reshaped into $X_2 \in \mathbb{R}^{HW\times C}$ and linearly projected into value $V \in \mathbb{R}^{HW\times C}$.

$$Q = X_1W^q, \quad K = X_1W^k, \quad V = X_2W^v \tag{3}$$

where W^q, W^k and $W^v \in \mathbb{R}^{C\times C}$ are the weights of 3 single-layer perceptrone. In a separate branch, X_{in} is reshaped as $X_3 \in \mathbb{R}^{C\times(HW)}$ and processed with 1D-convolution of kernel-size 1×1 to reduce the feature to $X_3 \in \mathbb{R}^{C_s\times(HW)}$. X_3 is again reshaped as $X_3 \in \mathbb{R}^{C_s\times H\times W}$ and processed by **"G-conv→GeLU→G-conv"** where **"G-conv"** stands for "*grouped-convolution*". Now the output is reshaped into *position-embedding* $E_{po} \in \mathbb{R}^{(HW)\times C_s}$.

Next, Q, K and V are subdivided along channel into h heads, such that each head $Q_i \in \mathbb{R}^{(HW/b^2)\times(C/h)}$, $K_i \in \mathbb{R}^{(HW/b^2)\times(C/h)}$ and $V_i \in \mathbb{R}^{(HW)\times(C/h)}$ where $i = 1, 2, ...h$. Now, self-attention is computed between the key and the query across the channel within each head. *E-SMSA* treats each channel as a token and computes attention among them within a head.

$$A_i = softmax(\sigma_i K_i^T Q_i), \quad A_i \in \mathbb{R}^{(C/h)\times(C/h)} \tag{4}$$

where $\sigma_i \in \mathbb{R}^1$ is a learnable parameter employed to adapt self-attention A_i during matrix-multiplication $K_i^T Q_i$. For single head $h = 1$, self-attention matrix $A_i \in \mathbb{R}^{C\times C}$ which is depicted in figure 6 for simplicity. Next, value V_i is matrix-multiplied with self-attention matrix A_i to generate a fused-feature H_i for i^{th} head.

$$H_i = V_i^T A_i, \quad H_i \in \mathbb{R}^{(HW)\times(C/h)} \tag{5}$$

The fused feature H_i from all h heads are now concatenated and linearly projected to generate *projection-embedding* E_{pr}.

$$E_{pr} = (concat(H_i))W, \quad E_{pr} \in \mathbb{R}^{(HW)\times C_s}, \quad i = 1, 2, ..., h \tag{6}$$

where $W \in \mathbb{R}^{C\times C_s}$ is the learnable weights of a single-layer perceptrone. Finally, the output of *E-SMSA* layer is computed as below.

$$X_{out} = E_{pr} + E_{po}, \quad X_{out} \in \mathbb{R}^{(HW)\times C_s} \tag{7}$$

Here, X_{out} is then reshaped into residual feature output F_s^r as mentioned in the section III-C.

IV. Network Implementation

A. Training/Validation Dataset Preparation

We train our proposed *TSF-Net* with Div2K [54] dataset, which includes 900 high-quality still images of 8-bit depth. Out of 900 images, we consider 875 images for training the *TSF-Net* and the remaining 25 for validating the network. To align the *TSF-Net* to operate in the YUV color space, we convert the 8-bit RGB images in the Div2K dataset to YUV 4:2:0 10-bit color format. Then, we encoded the YUV images with the VVC reference software, VTM-11.0, at four quantization-parameter (QP) rate points: 27, 32, 37 and 42 following the common test condition (CTC) under all-intra (AI) profile. However, we encoded the YUV images by disabling all the components (DBF, SAO, ALF(CCALF)) of the default VVC in-loop filter as the *TSF-Net* replaces the whole in-loop filter block in the VVC codec. In this article, we only consider the luma (Y) component for testing with *TSF-Net*. Therefore, for each of the 900 images we collected its VTM-11.0 encoded luma (Y) component and corresponding partition and residue information to construct training and validation patches.

B. Network Training

To improve the variance in the training sample, we generate training patches on the fly during network training. The training involves cropping 50 patches of size 256×256 from a luma image (I) and its co-located partition (P) and residue image (R). The strides of a window are computed to be random but to lie within a certain range, such that the extracted 50 patches cover a whole image but are also slightly different in the later epochs. During one epoch, the network sees a total of $875 \times 50 = 43750$ training patches. To augment the data, the training patches are randomly flipped horizontally and vertically. On the other hand, validation patches are prepared only once before network training. From 25 validation images, $25 \times 100 = 2500$ validation patches of size 256×256 are created with a fixed stride. Once the stride is fixed, the same set of validation patches is created for all sets of experiments.

Since the training and validation patch size is $H = W = 256$, the size of spatial input and frequency-decomposed input to *TSF-Net* is 128×128 and 64×64, respectively. In this article, we cascade $B = 20$ ***RFB*** blocks. All 2d-convolution operations involved in *TSF-Net* employ a kernel of 3×3. Similarly, we set the spatial feature channel size to $C_s = 96$ and frequency-decomposed feature channel size to $C_f = 32$. We set $h = 4$, so *E-SMSA* operates with 4 heads and the size of each self-attention matrix is $A_i \in \mathbb{R}^{32\times 32}$.

We train four separate models for four different QP values. However, we first train a model from scratch at $QP = 42$ utilizing training/validation patches encoded at the same QP. The models for the other QP values ($QP = \{37, 32, 27\}$) are then initialized with the weights from the pre-trained model at $QP = 42$ and fine-tuned with the patches encoded at the respective QP values. This approach allows the rest of the model to train faster and potentially with better performance. We adopt the Adam optimizer with $\beta = (0.9, 0.999)$ and *L1*-loss to optimize the network parameters. The model at

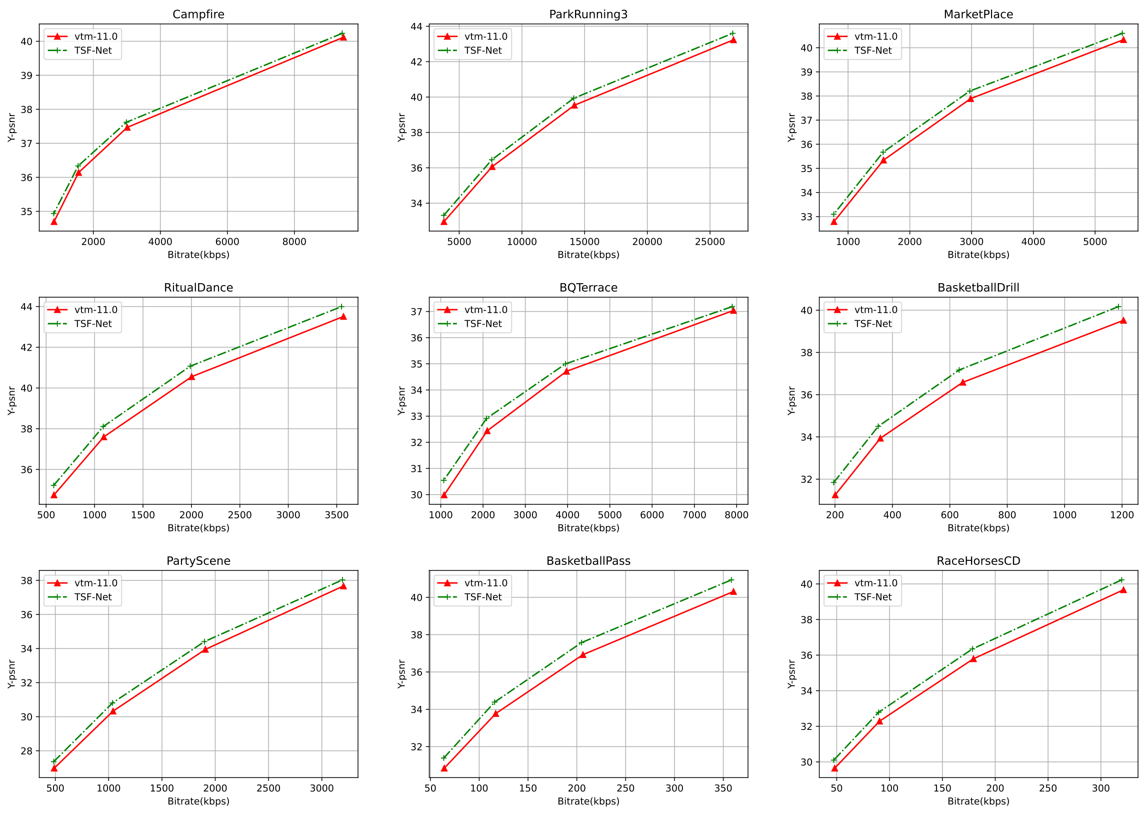

Fig. 7. Rate-Distortion (RD) plots for few test sequences comparing *TSF-Net* with VVC reference software. The Y-PSNR (dB) are evaluated at 4 QPs={42, 37, 32, 27}.

TABLE I
DECODING TIME COMPLEXITY OF VVC WITH TSF-NET (ON GPU) COMPARED TO VVC WITH ITS IN-LOOP FILTER ENABLED.

Class	**Decoding Complexity (%)**
A1	965.87
A2	713.11
B	1369.49
C	1987.32
D	4966.89
Overall	2089.54

$QP = 42$ is trained for 150 epochs, while the models for the other QP values ($QP = \{37, 32, 27\}$) are trained for only 100 epochs. The initial learning rate of the model at $QP = 42$ is set to 10^{-4}, while for the other models, it is initialized to 0.5×10^{-4}. We use a cosine-annealing learning-rate scheduler to decrease the learning rate until 10^{-6} during training. The batch-size is set to 16 for all model training.

V. EXPERIMENTAL RESULTS AND DISCUSSION

We evaluate the BD-rate performance of *TSF-Net* at four QPs={27, 32, 37, 42} against the VVC reference software, VTM-11.0 (with all in-loop filters enabled), under the all-intra (AI) configuration. To test *TSF-Net*, we disable all the components of the in-loop filter of VVC reference software and replace it with *TSF-Net*. We choose 19 common test sequences [56] recommended by *JVET* to evaluate the performance of *TSF-Net*. The selected 19 sequences are from 5 classes: A1 (3840×2160), A2 (3840×2160), B (1920×1080), C (480×832) and D (240×416). These test sequences are listed in Table II.

A. *Comparison of TSF-Net with VVC's In-loop Filter*

The rate-distortion (RD) plots (Bitrate vs 'Y'-PSNR) for a few test sequences are shown in figure 7. The RD-plots clearly indicate that the *TSF-Net*, as an in-loop filter, outperforms VTM-11.0's in-loop filter at all QP values. Moreover, except for a few sequences, the performance gap between VTM-11.0 and *TSF-Net* gets widens at lower QPs or correspondingly at larger bitrates. This suggests that our method tends to perform better at lower QPs compared to VVC. Blocking is less pronounced at lower bitrates and the distortions are primarily due to quantization noises. Therefore, the above observation points to the benefit of using frequency-decomposed input as prior information to the high-frequency content of an image. Here, *TSF-Net* is able to intercept high-frequency noises even in relatively good quality images (i.e. at lower bitrates) and correct them accordingly. This also demonstrates the powerful feature fusion capability of *E-SMSA* layer.

For example, figure 8 illustrates the difference in the reconstructed picture processed by VVC's in-loop filter and *TSF-Net*. The middle and right figures are reconstructed versions of the left figure, coded and reconstructed by VTM-11.0 at $QP = 32$. While the middle one is processed by VTM-11.0's

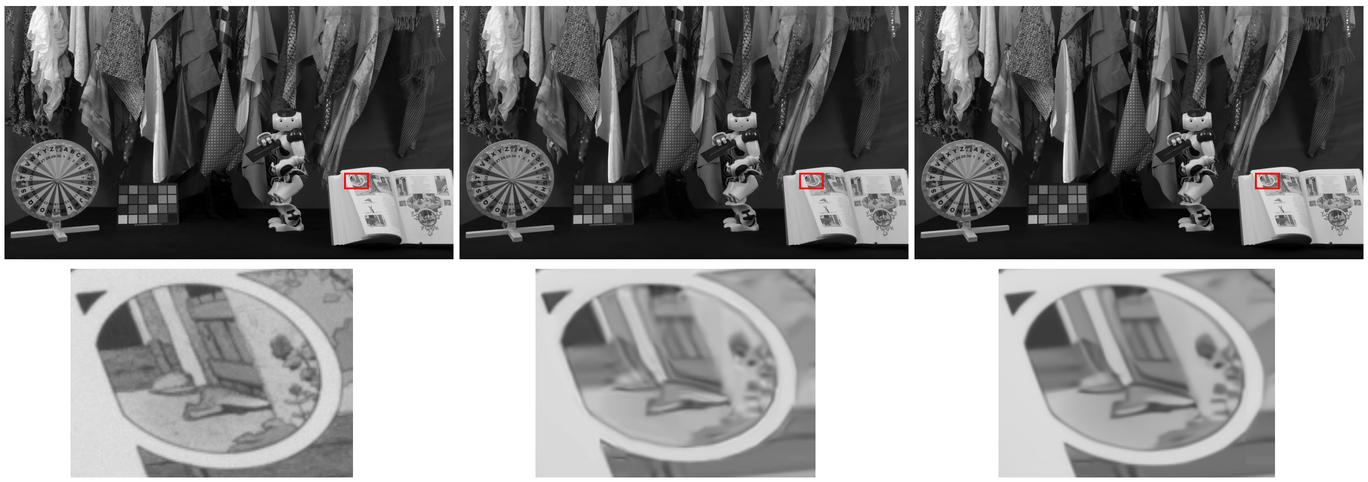

Fig. 8. *Left*: An original frame from CatRobot sequence. *Middle*: The same frame coded at QP=32 and processed with VTM-11.0's In-loop filters. *Right*: The same frame coded with VTM-11.0 at QP=32 and processed with *TSF-Net*.

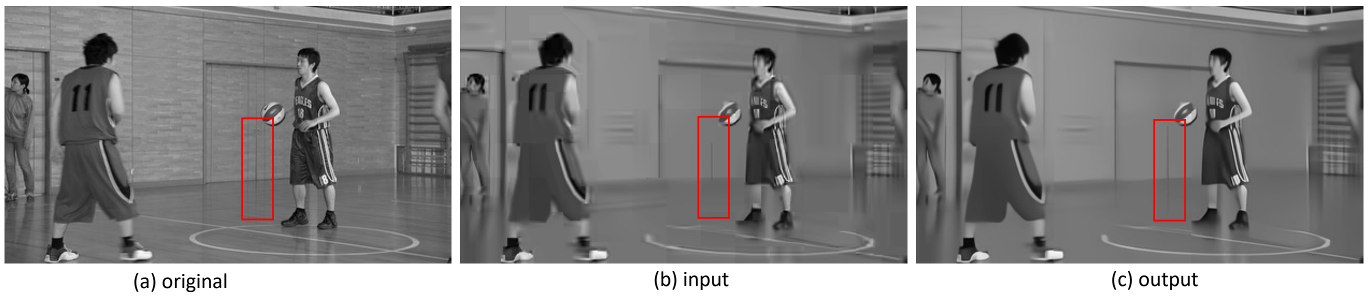

Fig. 9. *Left*: An original frame from BasketballPass sequence. *Middle*: The same frame coded and reconstructed at QP=47 by VTM-11.0 but yet to be processed by the in-loop filter. This is also an input to *TSF-Net*. *Right*: The output frame of *TSF-Net* after processing the frame shown in the middle.

in-loop filter, the right one is processed by the proposed *TSF-Net*. The zoomed-in pictures clearly demonstrate the difference between the two methods. The picture processed by VVC's in-loop filter exhibits ringing effects around the edges of the ring and appears slightly fuzzy. In contrast, the picture processed by *TSF-Net* has almost no ringing noises and looks sharper.

Similarly, figure 9 reveals an interesting aspect of *TSF-Net*: its slight generative capability. The leftmost figure is an original frame taken from *BasketballPass*. It is then coded at $QP = 47$ by VTM-11.0 (with in-loop filter disabled) and given as an input to *TSF-Net* which is shown in the middle figure. While the rightmost figure displays the processed output from *TSF-Net*. The vertical strip shown in the original figure, within the red rectangle, is shorter in the input frame and its reflection is absent on the floor. However, in the output frame generated by *TSF-Net*, the same strip is longer than in the input frame, and its reflection on the floor is appropriately generated. This confirms the generative capability of *TSF-Net*. This capability may be attributed to the fact that *TSF-Net* has learned to generate new information employing locality, or that the information was present in the input frame but not visible to the human eye, and *TSF-Net* made it more pronounced.

We also present the time complexity of *TSF-Net* as an in-loop filter in Table I. Since, *TSF-Net* is implemented only on the decoder side, the table includes the decoding time complexity against VVC when its in-loop filters are enabled. The decoding time complexity is calculated as $100 \times (t_{test}/t_{ref})$, where t_{test} is the decoding time of VVC with *TSF-Net* as in-loop filter and t_{ref} is the decoding time of VVC with its in-loop filter enabled. For each sequence, we obtain the final decoding time complexity by averaging them over all the QPs. Table I presents these values averaged over per class and all the sequences. From the table, we observe that our proposed *TSF-Net* also suffer the same fate as most other learning-based solutions, which is being slower. While analysing decoding time complexity per class, we notice that the complexity increases when video resolution decreases. One promising future research direction would be to improve the decoding time complexity.

B. Comparison of TSF-Net with other Learning-based In-Loop Filter

We also obtain the BD-rate and BD-PSNR performance of three additional learning-based methods: Deep In-loop Filter with Adaptive Model Selection (DAM) [55] [57], Uformer [14], and MSTFNet [51]. These methods were also evaluated against VTM-11.0, and their BD-rate and BD-PSNR performance, along with *TSF-Net*, for all 19 sequences can be found in Table II. Additionally, Table III presents the number of parameters (in million) and inference time (in seconds) for all four models. The inference time was tested on a Linux

TABLE II
COMPARISON OF BD-RATE(%) AND BD-PSNR(dB) PERFORMANCE ON LUMA COMPONENT UNDER ALL-INTRA (AI) CONFIGURATION

Class	Test Sequences	DAM [55]		Uformer [14]		MSTFNet [51]		TSF-Net	
		BD-BR	BD-PSNR	BD-BR	BD-PSNR	BD-BR	BD-PSNR	BD-BR	BD-PSNR
A1 (2160x3840)	Tango2	-10.015	0.241	-11.661	0.282	-11.099	0.267	-11.154	0.270
	FoodMarket4	-9.471	0.384	-10.162	0.414	-9.687	0.394	-9.843	0.401
	Campfire	-6.759	0.157	-7.640	0.176	-7.401	0.171	-7.799	0.180
A2 (2160x3840)	CatRobot	-11.290	0.408	-12.694	0.460	-11.910	0.430	-12.129	0.441
	DaylightRoad2	-12.220	0.313	-14.202	0.367	-13.519	0.348	-13.874	0.359
	ParkRunning3	-7.233	0.396	-7.042	0.384	-7.109	0.388	-7.365	0.403
B (1080x1920)	MarketPlace	-7.466	0.298	-8.268	0.332	-8.037	0.322	-8.212	0.330
	RitualDance	-10.366	0.529	-10.503	0.536	-10.320	0.526	-10.437	0.532
	Cactus	-9.301	0.349	-10.892	0.411	-10.476	0.396	-10.785	0.407
	BasketballDrive	-10.410	0.318	-11.066	0.339	-10.920	0.335	-11.025	0.338
	BQTerrace	-10.813	0.395	-10.485	0.383	-10.553	0.387	-10.741	0.392
C (480x832)	BasketballDrill	-13.571	0.673	-13.005	0.640	-13.740	0.683	-13.690	0.680
	BQMall	-11.285	0.627	-10.314	0.570	-11.364	0.631	-11.431	0.635
	PartyScene	-7.931	0.471	-7.482	0.443	-8.049	0.479	-7.974	0.474
	RaceHorses	-7.539	0.370	-7.093	0.348	-7.922	0.390	-8.064	0.397
D (240x416)	BasketballPass	-11.272	0.658	-10.040	0.583	-11.369	0.665	-11.347	0.663
	BQSquare	-10.021	0.694	-8.801	0.606	-10.00	0.693	-9.952	0.689
	BlowingBubbles	-8.862	0.498	-8.274	0.464	-9.07	0.511	-8.980	0.506
	RaceHorses	-10.013	0.557	-8.894	0.492	-10.081	0.561	-10.105	0.562
Average-Class A1		-8.748	0.261	-9.821	0.291	-9.396	0.277	-9.599	0.283
Average-Class A2		-10.248	0.372	-11.313	0.404	-10.846	0.389	-11.123	0.401
Average-Class B		-9.671	0.378	-10.243	0.400	-10.061	0.393	-10.240	0.400
Average-Class C		-10.081	0.535	-9.473	0.500	-10.269	0.546	-10.290	0.546
Average-Class D		-10.042	0.602	-9.002	0.537	-10.130	0.607	-10.096	0.605
Average-Overall		-9.781	0.439	-9.922	0.433	-10.138	0.451	-10.258	0.456

TABLE III
NO. OF PARAMETERS AND INFERENCE SPEED OF DIFFERENT MODELS

Model	no. of params (million)	inference time (second)
Bytedance	~ 5.51	0.0234
Uformer	~ 5.29	0.0182
MSTFNet	~ 25.77	0.0204
TSF-Net (vanilla)	~ 5.42	0.051
TSF-Net	~ 5.42	0.026

TABLE IV
PSNR PERFORMANCE OF TWO VARIANTS OF TSF-NET (VANILLA AND EFFICIENT) AT QP=42.

Class	TSF-Net (vanilla)	TSF-Net (efficient)
A1	36.482	36.486
A2	33.899	33.898
B	32.969	32.967
C	30.299	30.260
D	29.111	29.108
Average PSNR	32.296	32.293

system with PyTorch 2.0 and Nvidia RTX A6000 GPU and is the average time required to process an input image of size 256×256 over 100 rounds.

The Deep In-loop Filter with Adaptive Model Selection (DAM) was proposed by ByteDance to a JVET meeting as a response to the CfP on "neural network-based video coding" and has also been studied in [57]. DAM is placed before ALF while disabling DBF and SAO filters in VTM-11.0. During inference, patches are processed at the coding unit (CU) level, and for intra-coding, a CU-level flag is sent to indicate whether the CU-block is processed by DAM or VVC's default in-loop filter as a rate-distortion (RD) optimization. In contrast, there is currently no such optimization proposed with *TSF-Net*. During inference, *TSF-Net* processes high-resolution images at a patch size of 512×512, while low-resolution images (smaller than 512×512) are processed as a whole. Comparing the coding performance from Table II, we see that the BD-rate and BD-PSNR gain of our proposed *TSF-Net* is larger than that of *DAM* for all sequences (except BQTerrace). On average, *TSF-Net* outperforms *DAM* in all classes and leads

TABLE V
COMPARISON OF BD-RATE SAVING ON LUMA COMPONENT UNDER ALL-INTRA CONFIGURATION OF TSF-NET WHEN "FREQUENCY PATH" IS DISABLED.

Class	**TSF-Net (FPD)** (sallow)	**TSF-Net (FPD)** (deep)	**TSF-Net**
Average-Class A1	-8.956	-9.330	-9.599
Average-Class A2	-9.938	-10.586	-11.123
Average-Class B	-9.318	-9.908	-10.240
Average-Class C	-9.549	-10.177	-10.290
Average-Class D	-9.614	-10.061	-10.096
Average Overall	-9.470	-10.013	-10.258

by -0.477% and 0.017dB in BD-rate and BD-PSNR gain respectively. From Table III, we observe that both *TSF-Net* and *DAM* have approximately the same number of parameters and almost the same inference speed.

Considering the remarkable performance of *Uformer* [14] in various image restoration (IR) tasks such as image denoising, motion blur removal, defocus blur removal and rain removal, we investigate its potential as an in-loop filter and compare its performance with *TSF-Net*. *Uformer* is a transformer-based U-shaped IR model. We choose *Uformer-T (Tiny) (C = 16, depths of Encoder={2,2,2,2})* variant which has approximately 5.29 millions parameter, similar to *TSF-Net*'s parameters. We follow the training setup from [14] (for e.g. initialize learning rate to 2×10^{-4} and decrease it until 10^{-6}) to train a model at $QP = 42$. For other $QPs = \{37, 32, 27\}$ we follow the training strategy of *TSF-Net* (i.e. initialize the weights from model trained at $QP = 42$) while initializing the learning rate to 10^{-4} and decrease it until 10^{-6}. Analyzing the results in Table II, we observed that *Uformer* outperforms all other methods in *Class A1* and *Class A2*. However, in *Class C* and *Class D*, *Uformer* performs less effectively compared to other methods. In *Class B*, it achieves similar performance to *TSF-Net*. Overall, *Uformer* falls behind *TSF-Net* by 0.336% in terms of BD-rate and 0.023 dB in terms of BD-PSNR gain. Its inference speed is slightly better, nonetheless comparable to *TSF-Net* as presented in Table III. The performance analysis of *Uformer* emphasizes the strength of transformer-based models in exploiting long-range correlations in high-resolution videos, for example, in *Class A1* and *Class A2*, where CNN-based models lack severely. Conversely, for low-resolution videos such as *Class C* and *Class D*, CNN-based model proves to be the better choice.

We also consider the recently proposed *MSTFNet* [51] and compared it with *TSF-Net*. *TSF-Net* can be considered an improvement over *MSTFNet* as it is built by cascading only *RFB*s, each containing one *S-MSA*, resulting in a significantly higher number of feature-fusion operations (*S-MSA*) in *TSF-Net*. In contrast, *MSTFNet* has only three fusion blocks with *S-MSA* and the rest are sub-blocks without *S-MSA*. *MSTFNet* also includes very large numbers of convolutional layers compared to *TSF-Net*. Despite having significantly fewer parameters, approximately 5.4 million compared to *MSTFNet*'s 25.7 million, *TSF-Net* performs better than *MSTFNet* with a BD-rate and BD-PSNR gain of -0.119% and 0.005dB respectively. This performance boost can be attributed to *TSF-Net*'s more frequent use of feature-fusion operations (*S-MSA*) than *MSTFNet*. Despite having more *S-MSA* layers, *TSF-Net*'s inference time is still comparable to *MSTFNet* due to its efficient *S-MSA* (E-SMSA), as shown in Table III.

C. Ablation Study

1) Comparison of TSF-Net with efficient vs vanilla S-MSA: The proposed *TSF-Net* is efficient due to the incorporation of efficient S-MSA (*E-SMSA*) into the *residual fusion block (RFB)*. *E-SMSA* utilizes average-pooling (**"avg-pool"**) to reduce the feature size by half and computes the key (K) and query (Q) with dimensions of $K \in \mathbb{R}^{(HW/b^2)\times C}$ and $Q \in \mathbb{R}^{(HW/b^2)\times C}$, respectively, where $b = 2$. This reduces the computation complexity in the self-attention module by 1/4. In contrast, in the vanilla *TSF-Net*, average-pooling is absent, and K and Q are computed at the original size of $HW \times C$, making it less efficient.

Table IV displays the PSNR performance of the vanilla and efficient *TSF-Net* on JEVT sequences at $QP = 42$. From the table, we observe that there is barely any performance difference between the two models. However, it is worth noting that the proposed (efficient) *TSF-Net* is twice as efficient as the vanilla *TSF-Net*, as presented in table III. Please note that both models have the same number of parameters. Here, the PSNR performance is only presented at $QP = 42$, which we assume to be sufficient to demonstrate the advantage of the efficient *TSF-Net* over the vanilla *TSF-Net*. This is because the models at other QPs are initialized from the model trained at $QP = 42$ and are further trained to adapt them to other QPs. As a result, the PSNR performance at other QPs is expected to follow a similar pattern and hence is not included.

2) BD-Rate Performance of TSF-Net with no "Frequency-Decomposed Input": We also investigate the contribution of the "frequency-decomposed input" in the proposed TSF-Net. To do this, we remove the "frequency-decomposed input" and all the components responsible for processing it from the TSF-Net. In other words, we only keep the "spatial path" and disable the "frequency path". When the "frequency path" is disabled, the *Residual Fusion Block (RFB)* reduces to a simpler ResBlock (**"conv→ReLU→conv"**). We refer to this model as TSF-Net (FPD), where FPD stands for "Frequency Path Disabled". By removing the **"conv"** and **"T-conv"** layers from the "frequency path" and eliminating the need for the *E-SMSA* layer for feature fusion, the total number of trainable parameters in the TSF-Net decreases from approximately 5.42 million to around 3.34 million. This configuration is referred to as the "shallow" configuration. To ensure a fair comparison with the proposed TSF-Net, we increase the number of *RFB* blocks from 20 to 32 and disable the "frequency path" in the TSF-Net, resulting in another configuration of TSF-Net (FPD). This configuration of TSF-Net (FPD) has a total of approximately 5.34 million trainable parameters and is referred to as the "deep" configuration.

We train TSF-Net (FPD) in both the *sallow* and *deep* configurations at four QPs={27, 32, 37, 42}. The average BD-Rate savings per class on the Luma component under the all-intra (AI) profile for both configurations are presented in table V. We observe that the proposed TSF-Net offers greater BD-Rate savings compared to TSF-Net (FPD) in both configurations. The TSF-Net (FPD) with the *sallow* configuration, which has the smallest number of trainable parameters, achieves the lowest BD-Rate gain as expected. However, even with a similar number of trainable parameters to the proposed TSF-Net, the TSF-Net (FPD) in the *deep* configuration still demonstrates inferior BD-Rate performance compared to TSF-Net. These findings emphasize the significance of the "frequency-decomposed input" and the powerful feature fusion operation offered along the "frequency path" in the TSF-Net.

VI. Conclusion

In this article, we introduce a novel neural network-based in-loop filter for VVC, referred to as *TSF-Net*. Previous approaches mostly overlooked frequency-decomposed information when designing neural network-based in-loop filters. In contrast, *TSF-Net* learns jointly from both spatial (pixel) and frequency-decomposed (DCT'ed) information to effectively eliminate distortions in video frames. It utilizes a multi-level feature extraction and feature fusion strategy to enhance performance. However, we deviate from previous methods by designing a convolution-based simple feature extractor and an advanced feature-aggregator based on channel-wise transformer (*S-MSA*). These are combined into a block called *Residual Feature Fusion Block (RFB)*, which allows for scalability. Additionally, we propose an efficient channel-wise transformer (E-SMSA) that improves the efficiency of the vanilla *TSF-Net* by nearly a factor of two. We evaluate *TSF-Net* on 19 JVET common test sequences and achieve -10.258% BD-rate gain and 0.456dB BD-PSNR gain on the luma (Y) component under the all-intra (AI) configuration. Our proposed *TSF-Net* also outperforms other state-of-the-art (SOTA) methods under similar conditions.

Limitation and Future Work: The future direction of this work will be to test *TSF-Net* on the chroma components (Cb, Cr) and extend the work to other configurations: Low-Delay P (LP), Low-Delay B (LB) and Random-Access (RA). Additionally, introducing a rate-distortion (RD) optimization, as proposed in *DAM*, will certainly bring the extra coding gain. Another crucial direction to explore will be to make *E-SMSA* fusion-layer even more efficient and possibly make it shared among *RFB* blocks to reduce the number of parameters. Furthermore, *Uformer*'s results proved that *TSF-Net*'s performance is subpar in video sequence with larger resolution. This finding motivates to incorporate transformer as feature extractor in *TSF-Net* for future exploration.

Acknowledgments

This work is accomplished in collaboration with *Qualcomm* and partially supported by the NSF award 2148382.

References

[1] G. Sullivan and T. Wiegand, "Video compression - from concepts to the h.264/avc standard," *Proceedings of the IEEE*, vol. 93, no. 1, pp. 18–31, 2005.

[2] 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] 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.

[4] M. Karczewicz, N. Hu, J. Taquet, C.-Y. Chen, K. Misra, K. Andersson, P. Yin, T. Lu, E. François, and J. Chen, "Vvc in-loop filters," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 31, no. 10, pp. 3907–3925, 2021.

[5] A. Norkin, G. Bjøntegaard, A. Fuldseth, M. Narroschke, M. Ikeda, K. Andersson, M. Zhou, and G. V. der Auwera, "Hevc deblocking filter," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 22, pp. 1746–1754, 2012.

[6] C.-M. Fu, E. Alshina, A. Alshin, Y.-W. Huang, C.-Y. Chen, C.-Y. Tsai, C.-W. Hsu, S.-M. Lei, J.-H. Park, and W.-J. Han, "Sample adaptive offset in the hevc standard," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 22, no. 12, pp. 1755–1764, 2012.

[7] C.-Y. Tsai, C.-Y. Chen, T. Yamakage, I. S. Chong, Y.-W. Huang, C.-M. Fu, T. Itoh, T. Watanabe, T. Chujoh, M. Karczewicz, and S.-M. Lei, "Adaptive loop filtering for video coding," *IEEE Journal of Selected Topics in Signal Processing*, vol. 7, no. 6, pp. 934–945, 2013.

[8] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems*, F. Pereira, C. Burges, L. Bottou, and K. Weinberger, Eds., vol. 25. Curran Associates, Inc., 2012. [Online]. Available: https://proceedings.neurips.cc/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf

[9] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, "An image is worth 16x16 words: Transformers for image recognition at scale," in *International Conference on Learning Representations*, 2021. [Online]. Available: https://openreview.net/forum?id=YicbFdNTTy

[10] C. Dong, C. C. Loy, K. He, and X. Tang, "Image super-resolution using deep convolutional networks," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 38, no. 2, p. 295–307, feb 2016. [Online]. Available: https://doi.org/10.1109/TPAMI.2015.2439281

[11] D. Zhang, J. Shao, and H. T. Shen, "Kernel attention network for single image super-resolution," *ACM Trans. Multimedia Comput. Commun. Appl.*, vol. 16, no. 3, jul 2020. [Online]. Available: https://doi.org/10.1145/3398685

[12] Z. Lu, J. Li, H. Liu, C. Huang, L. Zhang, and T. Zeng, "Transformer for single image super-resolution," in *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. Los Alamitos, CA, USA: IEEE Computer Society, jun 2022, pp. 456–465. [Online]. Available: https://doi.ieeecomputersociety.org/10.1109/CVPRW56347.2022.00061

[13] Y. Zhang, Y. Tian, Y. Kong, B. Zhong, and Y. Fu, "Residual dense network for image restoration," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 43, no. 7, pp. 2480–2495, 2021.

[14] Z. Wang, X. Cun, J. Bao, W. Zhou, J. Liu, and H. Li, "Uformer: A general u-shaped transformer for image restoration," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2022, pp. 17 683–17 693.

[15] J. Liang, J. Cao, G. Sun, K. Zhang, L. Van Gool, and R. Timofte, "Swinir: Image restoration using swin transformer," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV) Workshops*, October 2021, pp. 1833–1844.

[16] S. Gu, Y. Li, L. Van Gool, and R. Timofte, "Self-guided network for fast image denoising," in *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019, pp. 2511–2520.

[17] S.-J. Cho, S.-W. Ji, J.-P. Hong, S.-W. Jung, and S.-J. Ko, "Rethinking coarse-to-fine approach in single image deblurring," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2021, pp. 4641–4650.

[18] H. Zhang, Y. Dai, H. Li, and P. Koniusz, "Deep stacked hierarchical multi-patch network for image deblurring," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019, pp. 5978–5986.

[19] P. Maharjan, L. Li, Z. Li, N. Xu, C. Ma, and Y. Li, "Improving extreme low-light image denoising via residual learning," in *2019 IEEE International Conference on Multimedia and Expo (ICME)*, 2019, pp. 916–921.

[20] S. W. Zamir, A. Arora, S. H. Khan, H. Munawar, F. S. Khan, M.-H. Yang, and L. Shao, "Learning enriched features for fast image restoration and enhancement," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022.

[21] J. Fang, H. Lin, X. Chen, and K. Zeng, "A hybrid network of cnn and transformer for lightweight image super-resolution," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, vol. 26, no. 7, June 2022, pp. 3142–3155.

[22] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

[23] W. Shi, F. Jiang, and D. Zhao, "Single image super-resolution with dilated convolution based multi-scale information learning inception module," in *2017 IEEE International Conference on Image Processing (ICIP)*, 2017, pp. 977–981.

[24] T. Guo, H. S. Mousavi, and V. Monga, "Adaptive transform domain image super-resolution via orthogonally regularized deep networks," *IEEE Transactions on Image Processing*, vol. 28, no. 9, pp. 4685–4700, sep 2019. [Online]. Available: https://doi.org/10.1109%2Ftip.2019.2913500

[25] J. Guo and H. Chao, "Building dual-domain representations for compression artifacts reduction," in *Computer Vision – ECCV 2016*, B. Leibe, J. Matas, N. Sebe, and M. Welling, Eds. Cham: Springer International Publishing, 2016, pp. 628–644.

[26] S. Herbreteau and C. Kervrann, "Dct2net: an interpretable shallow cnn for image denoising," 2021. [Online]. Available: https://arxiv.org/abs/2107.14803

[27] Z. Zhao, J. Zhang, S. Xu, Z. Lin, and H. Pfister, "Discrete cosine transform network for guided depth map super-resolution," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2022, pp. 5697–5707.

[28] J. Zhao, J. Xie, R. Xiong, S. Ma, T. Huang, and W. Gao, "Pyramid convolutional network for single image deraining," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2019, pp. 9–16.

[29] P. Liu, H. Zhang, K. Zhang, L. Lin, and W. Zuo, "Multi-level wavelet-cnn for image restoration," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2018, pp. 773–782.

[30] 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)*, 2020, pp. 783–787.

[31] L. Murn, S. Blasi, A. F. Smeaton, and M. Mrak, "Improved cnn-based learning of interpolation filters for low-complexity inter prediction in video coding," *IEEE Open Journal of Signal Processing*, vol. 2, pp. 453–465, 2021.

[32] J. Ballé, V. Laparra, and E. P. Simoncelli, "End-to-end optimized image compression," in *International Conference on Learning Representations*, 2017. [Online]. Available: https://openreview.net/forum?id=rJxdQ3jeg

[33] G. Lu, W. Ouyang, D. Xu, X. Zhang, C. Cai, and Z. Gao, "Dvc: An end-to-end deep video compression framework," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019, pp. 11 006–11 015.

[34] Y. Cai, J. Lin, Z. Lin, H. Wang, Y. Zhang, H. Pfister, R. Timofte, and L. Van Gool, "Mst++: Multi-stage spectral-wise transformer for efficient spectral reconstruction," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2022, pp. 745–755.

[35] T. Lu, F. Pu, P. Yin, T. Chen, and W. Husak, "Adaptive reshaping for next generation video codec," in *Applications of Digital Image Processing XLI*, A. G. Tescher, Ed., vol. 10752, International Society for Optics and Photonics. SPIE, 2018, p. 107520W. [Online]. Available: https://doi.org/10.1117/12.2320807

[36] K. Misra, F. Bossen, and A. Segall, "On cross component adaptive loop filter for video compression," in *2019 Picture Coding Symposium (PCS)*, 2019, pp. 1–5.

[37] W.-S. Park and M. Kim, "Cnn-based in-loop filtering for coding efficiency improvement," in *2016 IEEE 12th Image, Video, and Multidimensional Signal Processing Workshop (IVMSP)*, 2016, pp. 1–5.

[38] T. Li, M. Xu, R. Yang, and X. Tao, "A densenet based approach for multi-frame in-loop filter in hevc," in *2019 Data Compression Conference (DCC)*, 2019, pp. 270–279.

[39] D. Ding, L. Kong, G. Chen, Z. Liu, and Y. Fang, "A switchable deep learning approach for in-loop filtering in video coding," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 30, no. 7, pp. 1871–1887, 2020.

[40] S. Zhang, Z. Fan, N. Ling, and M. Jiang, "Recursive residual convolutional neural network- based in-loop filtering for intra frames," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 30, no. 7, pp. 1888–1900, 2020.

[41] M. Li and W. Ji, "Lightweight multiattention recursive residual cnn-based in-loop filter driven by neuron diversity," *IEEE Transactions on Circuits and Systems for Video Technology*, pp. 1–1, 2023.

[42] H. Huang, I. Schiopu, and A. Munteanu, "Frame-wise cnn-based filtering for intra-frame quality enhancement of hevc videos," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 31, no. 6, pp. 2100–2113, 2021.

[43] S. Chen, Z. Chen, Y. Wang, and S. Liu, "In-loop filter with dense residual convolutional neural network for vvc," in *2020 IEEE Conference on Multimedia Information Processing and Retrieval (MIPR)*, 2020, pp. 149–152.

[44] G. Huang, Z. Liu, and K. Q. Weinberger, "Densely connected convolutional networks," *CoRR*, vol. abs/1608.06993, 2016. [Online]. Available: http://arxiv.org/abs/1608.06993

[45] Z. Huang, Y. Li, and J. Sun, "Multi-gradient convolutional neural network based in-loop filter for vvc," in *2020 IEEE International Conference on Multimedia and Expo (ICME)*, 2020, pp. 1–6.

[46] D. Ma, F. Zhang, and D. R. Bull, "MFRNet: A new CNN architecture for post-processing and in-loop filtering," *IEEE Journal of Selected Topics in Signal Processing*, vol. 15, no. 2, pp. 378–387, feb 2021. [Online]. Available: https://doi.org/10.1109%2Fjstsp.2020.3043064

[47] Z. Huang, J. Sun, X. Guo, and M. Shang, "One-for-all: An efficient variable convolution neural network for in-loop filter of vvc," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 32, no. 4, pp. 2342–2355, 2022.

[48] S. Kuanar, D. Mahapatra, V. Athitsos, and K. R. Rao, "Gated fusion network for sao filter and inter frame prediction in versatile video coding," 2021.

[49] S. Liu, L. Wang, P. Wu, and H. Yang, "Jvet ahg report: Neural networks in video coding (ahg9)," 2018. [Online]. Available: https://jvet-experts.org/doc_end_user/documents/10_San%20Diego/wg11/JVET-J0009-v1.zip

[50] B. Kathariya, Z. Li, H. Wang, and G. Van Der Auwera, "Multi-stage locally and long-range correlated feature fusion for learned in-loop filter in vvc," in *2022 IEEE International Conference on Visual Communications and Image Processing (VCIP)*, 2022, pp. 1–5.

[51] B. Kathariya, Z. Li, H. Wang, and M. Coban, "Multi-stage spatial and frequency feature fusion using transformer in cnn-based in-loop filter for vvc," in *2022 Picture Coding Symposium (PCS)*, 2022, pp. 373–377.

[52] B. Lim, S. Son, H. Kim, S. Nah, and K. Mu Lee, "Enhanced deep residual networks for single image super-resolution," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, July 2017, pp. 136–144.

[53] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016, pp. 770–778.

[54] R. Timofte, S. Gu, J. Wu, L. Van Gool, L. Zhang, M.-H. Yang, M. Haris *et al.*, "Ntire 2018 challenge on single image super-resolution: Methods and results," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2018, pp. 852–863.

[55] Y. Li, L. Zhang, and K. Zhang, "Ahg11: Deep in-loop filter with adaptive model selection," in *document JVET-V0100, 22nd JVET meeting*, 2021.

[56] F. Bossen, J. Boyce, K. Suehring, X. Li, and V. Seregin, "Jvet common test conditions and software reference configurations for sdr video," 2018. [Online]. Available: https://jvet-experts.org/doc_end_user/documents/12_Macao/wg11/JVET-L1010-v1.zip

[57] Y. Li, L. Zhang, and K. Zhang, "Idam: Iteratively trained deep in-loop filter with adaptive model selection," *ACM Trans. Multimedia Comput. Commun. Appl.*, vol. 19, no. 1s, jan 2023. [Online]. Available: https://doi.org/10.1145/3529107

VII. Biography Section

Birendra Kathariya (Student Member, IEEE) received his B.E. degree in Electronics and Communication Engineering from Nepal Engineering College, Bhaktapur, Nepal in 2012, and his Master's degree from the University of Missouri-Kansas City (UMKC), Kansas City, MO, USA in 2017. He is presently pursuing a Ph.D. in Electrical Engineering from the University of Missouri-Kansas City (UMKC) and working in the Multimedia Computation and Communication (MCC) Lab at UMKC. He has also interned at various companies including MediaTek (San Jose, CA), Futurewei Technologies (San Jose, CA), Tencent America (Palo Alto, CA), and Qualcomm Technologies Inc. (San Diego, CA). His research interests include point cloud compression, image/video compression, deep-learning model compression in federated learning.

Zhu Li (Senior Member, IEEE) is a professor with the Dept of Computer Science Electrical Engineering, University of Missouri, Kansas City, and the director of NSF I/UCRC Center for Big Learning (CBL) at UMKC. He received his PhD in Electrical Computer Engineering from Northwestern University in 2004. He was AFRL summer faculty at the UAV Research Center, US Air Force Academy (USAFA), 2016-18, 2020-23. He was senior staff researcher with the Samsung Research America's Multimedia Standards Research Lab in Richardson, TX, 2012-2015, senior staff researcher with FutureWei Technology's Media Lab in Bridgewater, NJ, 2010 2012, assistant professor with the Dept of Computing, the Hong Kong Polytechnic University from 2008 to 2010, and a principal staff research engineer with the Multimedia Research Lab (MRL), Motorola Labs, from 2000 to 2008. His research interests include point cloud and light field compression, graph signal processing and deep learning in the next gen visual compression, image processing and understanding. He has 50+ issued or pending patents, 190+ publications in book chapters, journals, and conferences in these areas. He is an IEEE senior member, associate Editor-in-Chief for IEEE Trans on Circuits System for Video Tech, associated editor for IEEE Trans on Image Processing(2020), IEEE Trans.on Multimedia (2015-18), IEEE Trans on Circuits System for Video Technology(2016-19). He received the Best Paper Award at IEEE Int'l Conf on Multimedia Expo (ICME), Toronto, 2006, and IEEE Int'l Conf on Image Processing (ICIP), San Antonio, 2007.

Geert Van der Auwera (Senior Member, IEEE) received the Ph.D. degree in Electrical Engineering from Arizona State University, Tempe, AZ, USA, in 2007, and the Belgian MSEE degree from Vrije Universiteit Brussel (VUB), Brussels, Belgium, in 1997. Presently, he is a Director at Qualcomm Technologies Inc. in San Diego, CA, USA, in the Multimedia R&D & Standards group where he actively contributed to the standardization efforts of JCT-VC's High Efficiency Video Coding (HEVC) and JVET's Versatile Video Coding (VVC). Currently, he is participating in MPEG's Point Cloud Compression activity. Until Jan. 2011, he was with Samsung Electronics in Irvine, CA, USA. Until Dec. 2004, he was Scientific Advisor with IWT-Flanders, the Institute for the Promotion of Innovation by Science and Technology in Flanders, Belgium. In 2000, he joined IWT-Flanders after researching wavelet video coding at IMEC's Electronics and Information Processing Department (VUB-ETRO) in Brussels, Belgium. In 1998, his MSEE thesis on motion estimation in the wavelet domain received the Barco and IBM prizes by the Fund for Scientific Research of Flanders, Belgium. His research interests are video coding, point cloud compression, XR, video traffic and quality characterization, video streaming mechanisms and protocols.