IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 24 June 2023, accepted 26 July 2023, date of publication 31 July 2023, date of current version 8 August 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3300028

RESEARCH ARTICLE

Toward High-Quality Real-Time Video Denoising With Pseudo Temporal Fusion Network

KEI SHIBASAKI AND MASAAKI IKEHARA, (Senior Member, IEEE)
Faculty of Science and Technology, Department of Electrical and Information Engineering, Keio University, Yokohama, Kanagawa 223-8522, Japan
Corresponding author: Kei Shibasaki (shibasaki@tkhm.elec.keio.ac.jp)

This work was supported by Keio University.

ABSTRACT With the increasing availability of high-resolution video recording and streaming, there is a need for fast and high-quality video denoising methods that can handle high-resolution videos. However, many existing methods fail to achieve high-quality denoising performance and computationally effecient at the same time. This paper proposes a video denoising network, Pseudo Temporal Fusion Network (PTFN), that satisfies these requirements. PTFN adopts a new Pseudo Temporal Fusion (PTF) module that captures pseudo-temporal relationships between video frames in combination with the Temporal Shift Module. PTFN also adopts a modern ConvBlock paradigm that breaks away from the classical ConvBlock paradigm, contributing to denoising performance and computationally effecient. PTFN achieves better performance than existing video denoising methods in terms of both video quality and computational effeciency. Specifically, PTFN has only about 16.7% of the computational cost of existing lightweight methods, while it improves denoising performance. PTFN is also superior in terms of memory consumption. It can process 1080p videos with a GPU with 24 GB RAM. In addition, a lighter version (PTFN Half) can process 2K videos at high speed under the same conditions.

INDEX TERMS Deep learning, video denoising, light weight network, pseudo temporal fusion, modern ConvBlock.

I. INTRODUCTION

The technology related to photographic equipment and the performance of processors for images and videos is improving, making it possible to record or stream high-resolution videos. However, it is impossible to eliminate the noise generated by recording and streaming video, and it degrades the quality of the videos. Therefore, video denoising is essential for video quality, and there is a growing demand for denoising methods that can process high-resolution videos at high speed. [1], [2].

Video denoising methods can be broadly divided into "CPU-based methods" and "GPU/deep learning-based methods." CPU-based methods [3], [4] can process high-resolution videos because they do not consume large amounts of memory. However, they are slow because CPUs process denoising, and their performance is low compared to GPU/deep learning-based methods. GPU/deep learning-based methods [1], [2], [5], [6] are currently mainstream. They perform high-quality denoising quickly because the calculation is optimized for GPUs. However, many GPU/deep learning-based methods have the drawback that they cannot process high-resolution videos because they consume a large amount of GPU memory. Some methods can process high-resolution videos [1], [7], but these methods are not powerful enough. In addition, excellent quality methods [8], [9] often use huge networks, which slow down the video processing speed, limiting its practicality. Some GPU/deep learning-based methods [8], [9] use networks with Transformers [10], [11]. However, Transformer-based networks cannot process high-resolution video because the computational cost increases rapidly with image resolution. Transformer-based methods with smaller computational costs have also been proposed, but they cannot reduce the computational cost to process high-resolution videos.[1]

The associate editor coordinating the review of this manuscript and approving it for publication was Szidonia Lefkovits.

[1]For example, when using the NVIDIA RTX 3090 24GB, VRT [8] cannot process 128 × 128 videos due to insufficient memory capacity.

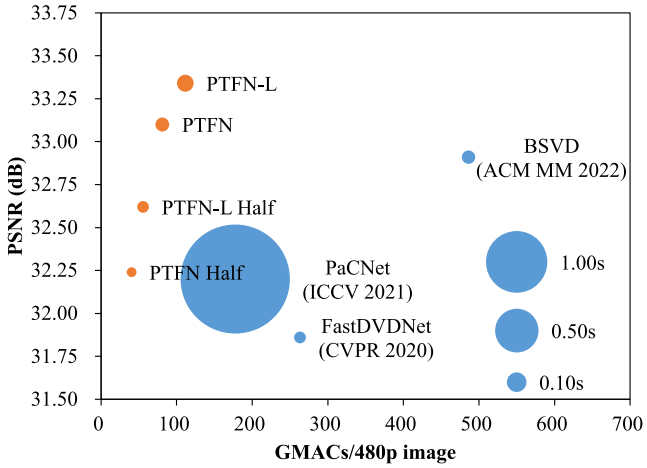

FIGURE 1. Comparison of the existing method (blue) and the proposed method (orange). The horizontal axis represents the computational cost (GMACs) per 480p image, the vertical axis represents the PSNR value for a noise level of 50 on the DAVIS test set, and the size of the bubble represents the processing time in PyTorch. The proposed method outperforms the existing lightweight method, BSVD, by only about 16.7% computational cost.

Recently, networks that re-examine the structure of classical CNNs and ConvBlocks [12], [13] (after this, referred to as ''modern CNNs/ConvBlock'') have achieved excellent results in the field of computer vision. Specifically, ''Classical CNN'' is the structure used in AlexNet [14] and VGG [15] and its derivatives, typically consisting of regular convolutions, batch normalization, and ReLU. ''Modern CNN'' is the structure used in ConvNeXt and its derivatives, with blocks consisting of Depthwise Separable Convolution, GELU, and Pointwise Convolution and blocks consisting of Pointwise Convolution and GELU. Each block is residually connected, and layer normalization is applied at the blocks' input. These structures are inspired by Transformer-based networks and are lighter and more powerful than Classical CNNs.

However, video denoising has not been as active as researches on image denoising, and many existing networks [1], [2], [5], [6] still use classical CNN.

This paper proposes a GPU/deep learning-based video denoising network, Pseudo Temporal Fusion Network (PTFN), achieving high-level denoising quality and computational effeciency simultaneously. PTFN uses a new Pseudo Temporal Fusion (PTF) module to capture temporal relationships between video frames. Since it does not process the temporal axis directly, it cannot capture temporal relationships by itself. However, when combined with Temporal Shift Module (TSM) [16], it can capture pseudo-temporal relationships. PTF contributes to performance and network's computational effeciency since it does not process the temporal axis directly, unlike heavy Conv3d. In addition, PTFN also uses modern ConvBlock, which contributes to both performance and computational effeciency. However, the ConvBlocks proposed in [12] and [13] are not verified to be suitable for video denoising, so this paper explores the structure of ConvBlock suitable for video denoising.

PTFN improves the denoising performance while significantly reducing the computational cost compared to existing methods. It also achieves excellent performance in terms of processing time in PyTorch [17]. PTFN is also superior in terms of memory consumption. It can rapidly process 1080p (1980×1080) videos with NVIDIA RTX 3090 24GB. In addition, a lighter version (PTFN Half) can process 2K videos at high speed under the same conditions.

The main contributions of this paper are as follows:

- We propose Pseudo Temporal Fusion Network (PTFN), a deep learning network that combines computational effeciency and denoising quality at a high-level. PTFN adopts a modern ConvBlock structure to improve computational effeciency and denoising quality. In addition, we propose Pseudo Temporal Fusion (PTF), a module for capturing the pseudo-temporal relationship between video frames. Since PTF does not directly calculate the temporal axis, it contributes to computational effeciency of the network.
- PTFN achieves excellent video denoising performance. PTFN improves the PSNR value by 0.19 dB on the DAVIS test set [18] with a noise level of 50 compared to BSVD [2], the existing lightweight video denoising method. Moreover, PTFN is a very computationally effecient network with only 16.7% computational cost from BSVD.

II. RELATED WORKS

A. VIDEO DENOISING

1) CPU-BASED METHODS

Video denoising has long been discussed, and many researches are tackling it before GPU/deep learning-based methods appear. Maggioni et al. [4] proposed a filtering algorithm that exploits the temporal and spatial redundancy that characterizes natural video. This method exploits temporal and spatial correlations through non-local grouping. Jovanov et al. [19] proposed a selective wavelet shrinkage algorithm with motion estimation and noise level estimation. Arias and Morel [3] proposed a method to build Bayesian models for similar spatio-temporal patches.

These methods are too slow because the computational processing is not optimized for GPUs. They also have lower performance than the GPU/deep learning-based methods that have recently become mainstream.

2) GPU/DEEP LEARNING-BASED METHODS

Recently, GPU/deep learning-based methods have been the mainstream of video denoising because of fast inference and high performance. Tassano et al. [5] proposed a network that performs temporal denoising after spatial denoising and achieves performance that exceeds existing CPU-based methods. Since [5], many GPU/deep learning-based video denoising methods have been proposed, and the denoising performance has been dramatically improved.

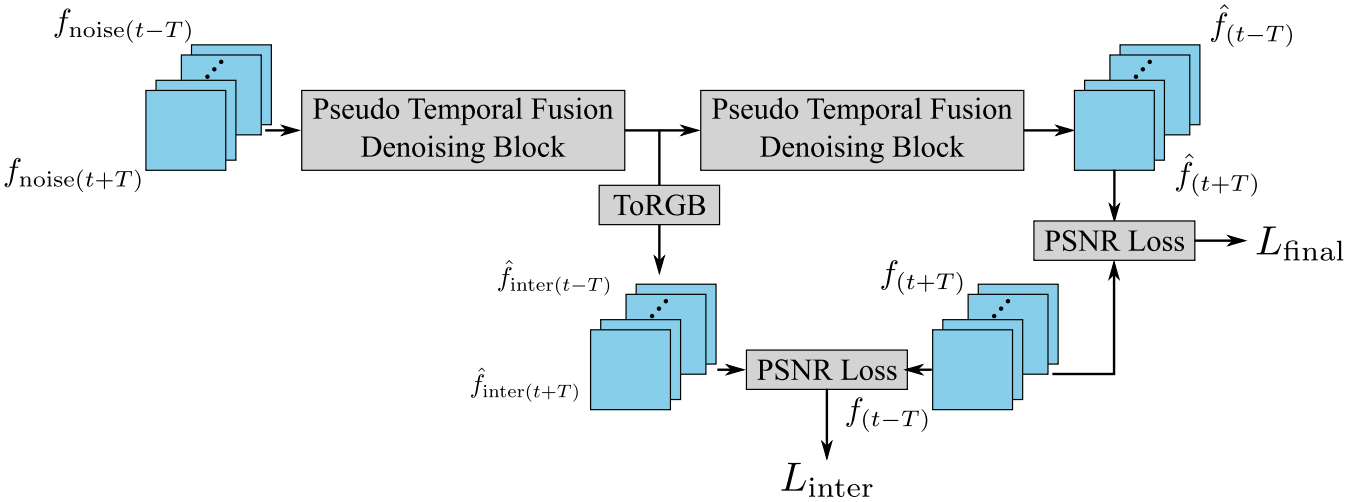

FIGURE 2. The overall structure of Pseudo Temporal Fusion Network (PTFN).

Recently, Vaksman et al. [6] proposed a method to introduce patch-craft frames, pseudo-frames similar to actual frames, to augment video sequences and denoise them in CNNs. Liang et al. [8] proposed a module that recognized the correlation between frames using the Attention module. Buades and Lisani [20] proposed a method to capture temporal relationships using a regularized optical flow method. Shen et al. [21] proposed a complementary network based on optical flow for spatial enhancement and flow enhancement modules for temporal enhancement. Liang et al. [9] proposed a method that efficiently exploits temporal relationships using Guided Deformable Attention. Although [8] and [9] achieved excellent performance, they are computationally very expensive due to the use of Transformers in the network. In addition, the inference speed of these methods is also slow due to the huge size of the network. It is essential to process high-resolution videos rapidly for practical use, so a computationally effecient network with high denoising performance is required.

Several lightweight networks [1], [2], [7] for video denoisng have been proposed. Tassano et al. [1] proposed a fast denoising network consisting of two lightweight U-Nets [22]. This method combines frames in the channel direction. It fuses the relationship between frames and channels to achieve fast inference, but it has the problem of inefficient inference because it performs sliding-window inference. Qi et al. [2] proposed a Bidirectional Buffer Block (BBB) that can replace the Temporal Shift Module (TSM) [16] and proposed a MIMO (Multi Input Multi Output) network that uses it. In [2], the performance degradation caused by temporal clipping, a popular problem of MIMO networks, is addressed by replacing the TSM with the BBB during inference. In these methods, the network structure is a classical CNN network, and there is room for improvement.

Unlike existing methods, the proposed method employs a modern ConvBlock, contributing computational effeciency of the network while keeping great performance. We propose a new Pseudo Temporal Fusion (PTF) module, which captures pseudo-temporal relationships of video sequences when combined with the Temporal Shift Module. PTF is suitable for computationally effecient video denoising networks compared to existing modules for capturing temporal relationships because it does not directly process the temporal axis.

B. MODERN CNN NETWORKS

Recently, image and video processing networks based on Transformer have achieved excellent results [9], [10], [11], [23]. Recently, however, some researches have begun to re-examine the structure of traditional CNNs regarding the structure of the Transformer networks. Liu et al. [12] reviewed the structure of ResNet-50 [24] regarding the Swin Transformer [11]. They proposed a new network called ConvNeXt. [12] modified ResNet-50 from five features. The features are "Macro design," "ResNeXt-ify [25]," "Reversed bottleneck," "Large kernel size," and "Micro design." The proposed ConvNeXt outperforms the Swin Transformer in COCO object detection and ADE-20K segmentation. Chen et al. [13] proposed a CNN network for image restoration tasks such as super-resolution and blur removal for a single image, referring to [12].

III. PSEUDO TEMPORAL FUSION NETWORK (PTFN)

A. OVERALL STRUCTURE

PTFN is a MIMO (Multi Input Multi Output) network with the structure shown in Fig. 2. Following some existing methods [1], [2], PTFN is a network consisting of two Pseudo Temporal Fusion Denoising Blocks (PTF Denoising Block) (Fig. 3). The input of PTFN is the noisy video sequence $[f_{\text{noise}(t-T)}, \cdots, f_{\text{noise}(t+T)}]$. The output of PTFN is denoised video sequence $[\hat{f}_{(t-T)}, \cdots, \hat{f}_{(t+T)}]$ and intermediate output $[\hat{f}_{\text{inter}(t-T)}, \cdots, \hat{f}_{\text{inter}(t+T)}]$. 1×1 Conv2d (ToRGB in Fig. 2) is applied before outputting intermediate output.

PTFN performs step-by-step denoising by connecting denoising networks in series, and there is a possibility that performance can be further improved by connecting three or more networks. However, since this paper discusses lightweight networks, we do not examine this possibility. Actually, if three are connected in series, the NVIDIA RTX 3090 24GB will not be able to process 1080p resolution videos.

B. PSEUDO TEMPORAL FUSION DENOISING BLOCK (PTF DENOISING BLOCK)

The PTF Denoising Block is a four-scale U-shaped network shown in Fig. 3. At each scale, the PTF Denoising Block processes video in layers consisting of the Temporal Shift Module (TSM), ConvBlock, and Pseudo Temporal Fusion Block (PTF Block). Downsampling is performed by 2×2 Conv2d with stride 2, and upsampling is performed by 1×1 Conv2d and Pixel Shuffle [26]. Unlike original U-Net, when fusing features, the features are not concatenated but are added element-wise. The number of channels is 32, 64, 128, and 256 from the above scale. Following [2], TSM is replaced with Bidirectional Buffer Block during inference.

C. TEMPORAL SHIFT MODULE (TSM)

TSM [16] is a module for fusing temporal information with channel information in video processing MIMO networks. TSM shifts specific ratio channels in the feature map to the previous and next frames. When inputting feature map f whose size is (B, T, C, H, W), the output of TSM f' is presented as:

$$f'[:,t,:,:,:] = \mathrm{Concat}\left(f\left[:,t-1,:\frac{c}{2r},:,:\right], f\left[:,t+1,\frac{c}{2r}:\frac{c}{r},:,:\right], f\left[:,t,\frac{c}{r}:,:,:\right]\right) \tag{1}$$

Note that B is the batch size, T is the number of frames, C is the number of channels, H is height, W is width, and r is the ratio to shift feature maps. $\mathrm{Concat}(x_1, x_2, \cdots, x_n)$ means to concatenate $[x_1, x_2, \cdots, x_n]$ to channel axis. All r are set to 8 in PTFN.

D. ConvBlock

PTFN adopts a modern ConvBlock structure, as shown in Fig. 4 (a). The ConvBlock is unlike the ConvBlocks used in existing networks, composed of Layer Normalization [27], 1×1 Conv2d, 3×3 Depthwise Conv2d [28], GELU [29], and 1×1 Conv2d. A residual connection between the input and output is adopted. The width of channels is twiced by 1×1 Conv2d before GELU and halved by 1×1 Conv2d after GELU.

The main difference between PTF Block and ConvBlock is the role in network processing. ConvBlock has spatial operations such as 3×3 DConv, but PTF Block does not have such operations. ConvBlock has the role of spatial processing, which is the difference from PTF Block.

E. PSEUDO TEMPORAL FUSION BLOCK (PTF BLOCK)

The structure of the PTF Block is shown in Fig. 4 (b). It consists of Layer Normalization, 1×1 Conv2d, Pseudo Temporal Fusion (PTF), and 1×1 Conv2d. The specific operations of PTF are calculated in (2) and (3).

$$X_{t-1}, X_t, X_{t+1} = \mathrm{Split}(X, 3) \tag{2}$$

$$Y = \mathrm{Like}(0.5, X_t) \otimes ((X_{t-1} \otimes X_t) \oplus (X_t \otimes X_{t+1})) \tag{3}$$

Note that $\mathrm{Split}(X, n)$ is the operation that splits X into n equal parts by channel axis, and $\mathrm{Like}(a, X)$ means a tensor of identical size to X with all values a. $\otimes$ and $\oplus$ means element-wise multiplication and element-wise addition. We use PyTorch in our experiments and torch.chunk is used for dividing the feature into equally three parts. The PTF divides the input into three equal parts by channel axis to separate temporal information from feature maps. Each divided feature map means the information at time t and before and after. Usually, it is impossible to separate temporal information by splitting feature maps in the channel axis. Thus, PTF cannot capture the temporal relationship alone. However, the PTF Denoising Block contains a TSM, and the temporal relationships are merged with the channel relationships. Therefore, PTF can capture pseudo-temporal relationships by splitting feature maps by channel axis between feature maps. PTF is also computationally effecient because it does not perform calculations directly on the temporal axis. Conv3d performs operations on the temporal axis in addition to the channel, horizontal and vertical axis in spatial. The PTF operation does not perform operations on the temporal axis. The ratio of the computational cost of PTF Block to that of 3×3 Conv3d is expressed by (4).

$$\frac{\underbrace{\frac{3hwc^2}{1 \times 1\ \mathrm{Conv2d}} + \underbrace{3hwc}_{\mathrm{PTF}} + \underbrace{\frac{hwc^2}{1 \times 1\ \mathrm{Conv2d}}}}}{\underbrace{9fhwc^2}_{3 \times 3\ \mathrm{Conv3d}}} \sim \frac{4}{9f} \tag{4}$$

Note that h, w, c, f represent the height, width, number of channels, and number of frames of the feature map. Eq. (4) indicates that Conv3d directly performs the temporal axis calculation, which increases the computational complexity over PTF by the number of frames f.

F. LOSS FUNCTION

The loss function used during training is expressed by:

$$\frac{1}{(2T+1)(1+\alpha)}\left(\sum_{\tau=t-T}^{t+T} \mathrm{PSNRLoss}(\hat{f}_\tau, f_\tau) + \alpha \sum_{\tau=t-T}^{t+T} \mathrm{PSNRLoss}(\hat{f}_{\mathrm{inter}(\tau)}, f_\tau)\right) \tag{5}$$

The meanings of the symbols in (5) follow those in Fig. 2, where t and T are the frame indices, $\hat{f}$ is the final output image of the network, $\hat{f}_{\mathrm{inter}}$ is the intermediate output of the network, and f is the clean image. $\mathrm{PSNRLoss}(x, y)$ means the

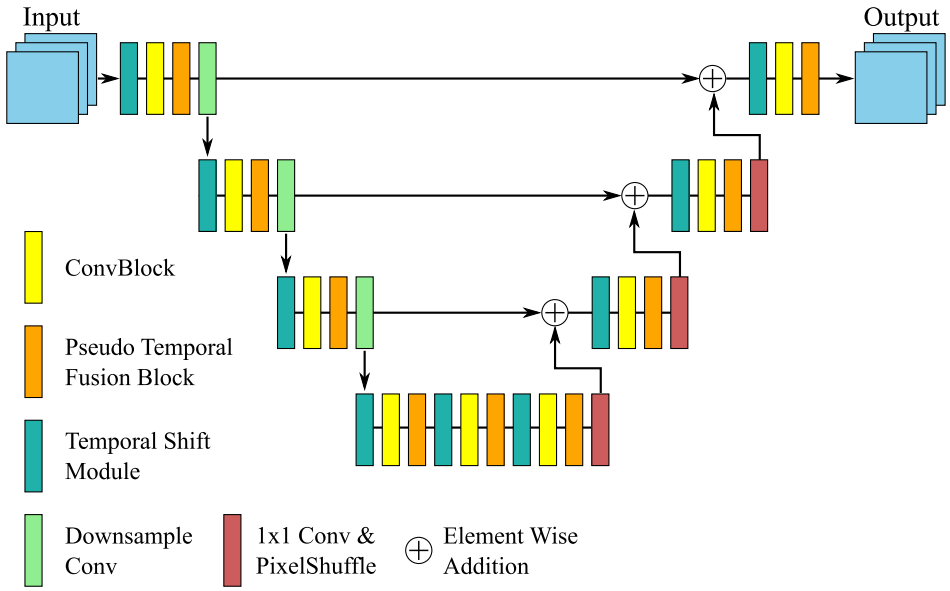

FIGURE 3. The structure of Pseudo Temporal Fusion Denoising Block (PTF Denoising Block).

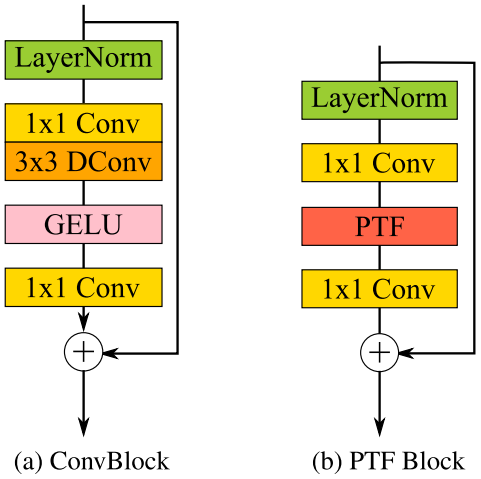

FIGURE 4. The structure of ConvBlock and Pseudo Temporal Fusion (PTF Block).

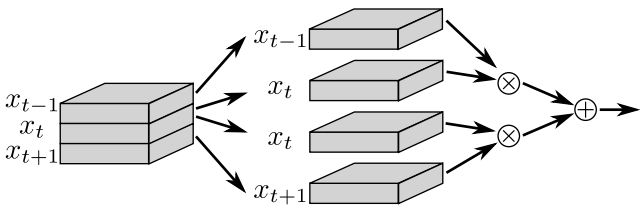

FIGURE 5. The structure of Pseudo Temporal Fusion. ⊗ and ⊕ means element-wise multiplication and element-wise addition.

PSNRLoss between x and y, and α is a coefficient to adjust the loss ratio.

Since many lightweight methods [1], [2] consist of two denoising networks connected in series, multi-stage denoising would be helpful for high-quality denoising. Therefore, in (5), the loss between the intermediate output and the clean image is adopted to assist in learning multi-step denoising. Since the PSNRLoss used in this training has a large absolute value, the value of the loss function changes significantly by adding a term, and the value of the gradient also changes significantly. Therefore, by dividing the loss value by $1 + \alpha$, the change in loss value caused by the value of α is reduced.

IV. EXPERIMENTS

A. DATASET

In order to compare the proposed method with existing methods, we follow [1] and perform the quantitative and qualitative comparison using the DAVIS data set [18] and the Set8 test set [1], [30]. The DAVIS dataset consists of a training set containing 90 RGB video sequences and a test set containing 30 RGB video sequences with a resolution of 480p (480 × 854). The Set8 test set consists of 4 video sequences from the Derf test set [30] and 4 video sequences captured by GOPRO with a resolution of 540p (540 × 920).

B. CONDITIONS

The batch size of the proposed method during training is 16. The number of input frames is 11, $T = 5$ according to the notation in Fig. 2. The input video is randomly cropped to 96 × 96, and random flipping is applied to each video sequence. Additive White Gaussian Noise (AWGN) is added to the video sequences, and the noise level σ is selected from the uniform distribution $U(5, 55)$.

The training is performed on the DAVIS train set with 400,000 iterations. Adam [31] with $\beta_1 = 0.9$, $\beta_2 = 0.9$ is used as the optimizer. Cosine Annealing [32] with $\eta_{max} = 1.0 \times 10^{-3}$, $\eta_{min} = 1.0 \times 10^{-7}$, and $T_{max} = 400,000$ is used as the learning rate scheduler. The ratio α of the intermediate output loss in the loss function (5) is set to 0.1. The value of the gradient is clipped at 0.1 during training.

TABLE 1. PSNR (dB) for the non-blind denoising on DAVIS and Set8 test sets. σ represents the AWGN noise level and Avg is the average PSNR for each noise level. The best value for each item is bolded, and the second best value is underlined.

	DAVIS PSNR (dB)						Set8 PSNR (dB)					
Method	$\sigma = 10$	$\sigma = 20$	$\sigma = 30$	$\sigma = 40$	$\sigma = 50$	Avg	$\sigma = 10$	$\sigma = 20$	$\sigma = 30$	$\sigma = 40$	$\sigma = 50$	Avg
VNLB [3]	38.85	35.68	33.73	32.32	31.13	34.34	**37.26**	33.72	31.74	30.39	29.21	32.46
V-BM4D [4]	37.58	33.88	31.65	30.05	28.80	32.39	36.05	32.19	30.00	28.48	27.33	30.81
DVDNet [5]	38.13	35.70	34.08	32.86	31.85	34.52	36.08	33.49	31.79	30.55	29.56	32.29
FastDVDNet [1]	38.71	35.77	34.04	32.82	31.86	34.64	36.44	33.43	31.68	30.46	29.53	32.31
PaCNet [6]	**39.97**	36.82	34.79	33.34	32.20	35.42	37.06	33.94	32.05	30.70	29.66	32.68
BSVD [2]	39.81	36.82	35.09	33.86	32.91	35.70	36.74	33.83	32.14	30.97	30.06	32.75
PTFN Half	39.18	36.11	34.37	33.16	32.24	35.01	36.35	33.42	31.72	30.54	29.64	32.33
PTFN	39.72	36.86	35.20	34.02	33.10	35.78	36.68	33.85	32.21	31.06	30.18	32.80
PTFN-L Half	39.44	36.48	34.77	33.56	32.62	35.37	36.52	33.64	31.97	30.81	29.91	32.57
PTFN-L	39.86	**37.05**	**35.41**	**34.24**	**33.34**	**35.98**	36.82	**33.99**	**32.36**	**31.22**	**30.34**	**32.95**

TABLE 2. PSNR (dB) for the blind denoising on DAVIS and Set8 test sets. σ represents the AWGN noise level and Avg is the average PSNR for each noise level. The best value for each item is bolded, and the second best value is underlined.

	DAVIS PSNR (dB)						Set8 PSNR (dB)					
Method	$\sigma = 10$	$\sigma = 20$	$\sigma = 30$	$\sigma = 40$	$\sigma = 50$	Avg	$\sigma = 10$	$\sigma = 20$	$\sigma = 30$	$\sigma = 40$	$\sigma = 50$	Avg
ReMoNet [7]	38.97	35.77	33.93	32.64	31.65	34.59	36.29	33.34	31.59	30.37	29.44	32.21
BSVD-blind [2]	39.68	36.66	34.91	33.68	32.72	35.53	36.54	33.70	32.02	30.85	29.95	32.61
PTFN-blind Half	39.25	36.30	34.59	33.38	32.44	35.19	36.27	33.47	31.83	30.67	29.78	32.40
PTFN-blind	39.65	36.76	35.08	33.90	32.98	35.67	36.58	33.78	32.14	30.99	30.11	32.72
PTFN-L-blind Half	39.40	36.46	34.76	33.56	32.62	35.36	36.36	33.57	31.94	30.78	29.89	32.51
PTFN-L-blind	**39.79**	**36.96**	**35.31**	**34.14**	**33.22**	**35.88**	**36.64**	**33.89**	**32.28**	**31.14**	**30.26**	**32.84**

TABLE 3. Comparison of the lightness of the models of the proposed and existing methods. Runtimes (s/image) represents the processing time per image in Python or PyTorch implementation, and GMACs/image represents the computational cost per image. 480p means 480 × 854, 720p means 720 × 1280, and 1080p means 1080 × 1920. For processing, the GPU-based method used an NVIDIA RTX 3090 24GB, and the CPU-based method used an Intel Xeon CPU E5-1650 v4. The PSNR values at $\sigma = 50$ for the DAVIS test set are also shown in Tab 3 to compare the trade-off between processing time and performance. OOM stands for Out Of Memory, and the computational cost per image is calculated only for the GPU-based method.

		PSNR (dB)	Runtimes (s/image)			GMACs/image	
Method	Hardware	$\sigma = 50$	480p	720p	1080p	480p	720p
VNLB [3]	CPU	31.13	128.97	264.18	583.13	N/A	N/A
V-BM4D [4]	CPU	28.80	151.34	335.81	763.12	N/A	N/A
DVDNet [5]	CPU+GPU	31.85	3.71	6.31	OOM	N/A	N/A
FastDVDNet [1]	GPU	31.86	0.037	0.081	0.18	263.20	498.11
PaCNet [6]	GPU	32.20	3.16	OOM	OOM	177.42	OOM
BSVD [2]	GPU	32.83	0.047	0.10	OOM	486.35	1090.88
PTFN Half	GPU	32.24	0.026	0.056	0.13	40.60	91.06
PTFN	GPU	33.10	0.051	0.11	0.25	81.16	182.03
PTFN-L Half	GPU	32.62	0.037	0.082	0.18	55.74	125.04
PTFN-L	GPU	33.34	0.074	0.16	0.37	111.45	249.98

In addition to the model using the PTF Denoising Block (PTFN) shown in Fig. 3, we also train PTFN-L in which the number of PTF Blocks at each stage of the PTF Denoising Block is increased from one to two. Lighter models with only one PTF Denoisig Block (PTFN Half and PTFN-L Half) are also trained. These models are fine-tuned using weights of trained PTFN and PTFN-L. The DAVIS train set is used for fine-tuning with 100,000 iterations. Adam is used as the optimizer with $\beta_1 = 0.9$ and $\beta_2 = 0.9$. Cosine Annealing with $\eta_{max} = 1.0 \times 10^{-4}$, $\eta_{min} = 1.0 \times 10^{-7}$, and $T_{max} = 100,000$ is used as the learning rate scheduler.

In addition, experiments were also conducted on blind video denoising. In this case, the training conditions are the same as non-blind denoising.

C. QUANTITATIVE COMPARISON

1) QUALITY OF GENERATED VIDEO

Tab. 1 compares the PSNR values of the non-blind denoising results on the DAVIS and Set8 test sets for the proposed and existing methods. Tab. 1 shows that our method achieves better performance for a wide range of noise levels than the existing methods. Compared to BSVD, PTFN and PTFN-L improve by 0.19 dB and 0.43 dB on the DAVIS test set and by 0.05 dB and 0.20 dB on the Set8 test set at a noise level of 50. Compared to FastDVDNet, PTFN Half and PTFN-L Half improved by 0.38 dB and 0.75 dB in the DAVIS test set and by 0.10 dB and 0.38 dB in the Set8 test set at a noise level of 50.

VNLB and PaCNet perform very well when $\sigma = 10$, and the proposed methods are not as good as these methods. These methods are patch-based, and they obtain useful features from the surroundings by patch matching. When σ is small, useful features can be obtained from surrounding patches because the image or video is relatively clean. Therefore, these conventional methods tend to perform better when σ is small. However, this tendency is limited to where σ is small, and our method performs much better when σ is high.

Tab. 2 compares the PSNR values of the blind denoising results on the DAVIS and Set8 test sets for the proposed and existing methods. As in the non-blind case, PTFN achieves

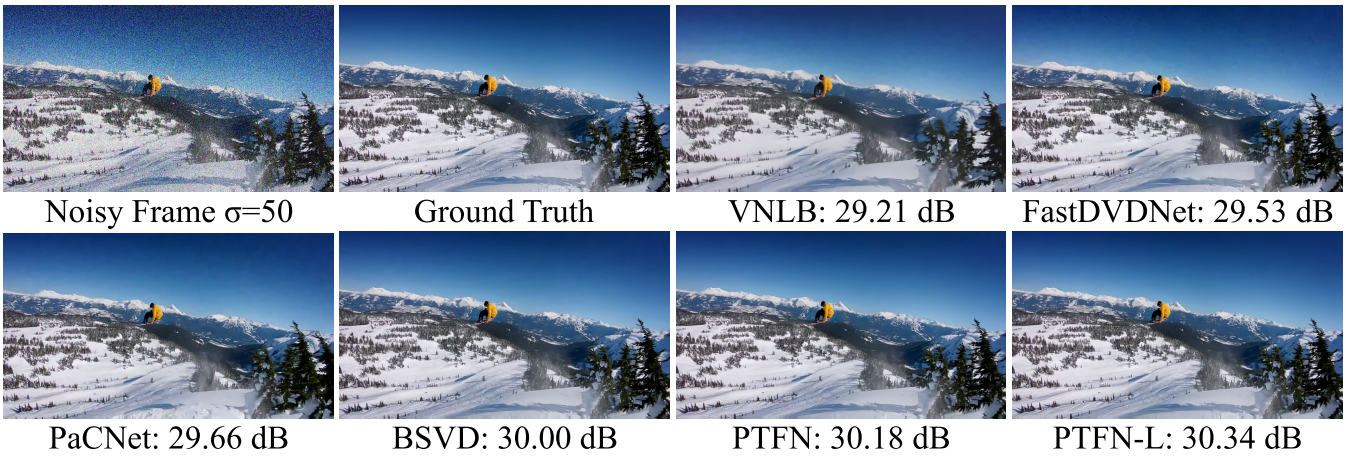

FIGURE 6. Comparison of the denoised images of the proposed method and the exiting method in one frame of video Snowboard in the Set8 test set.

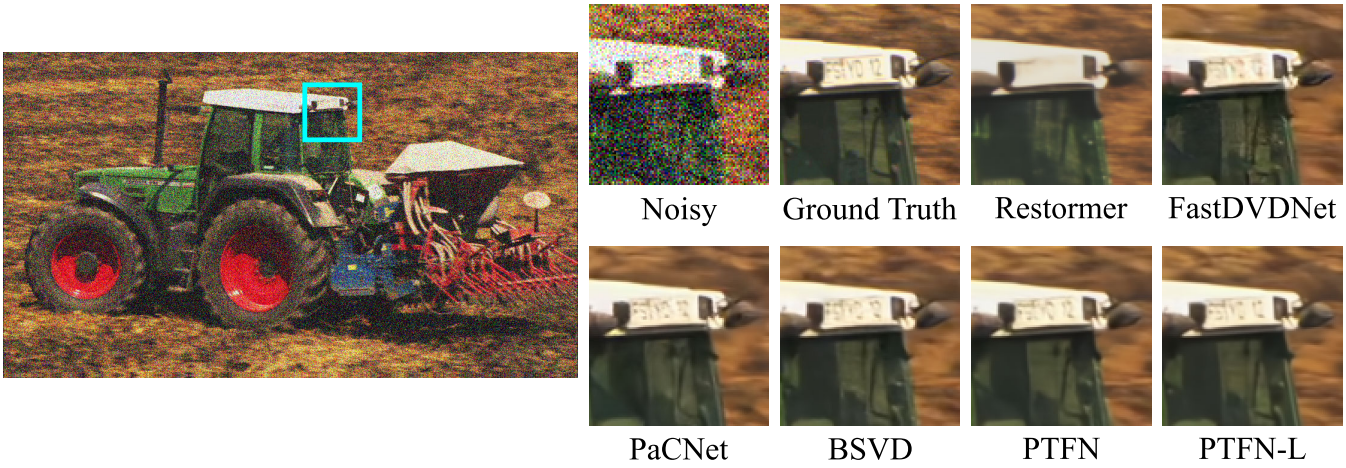

FIGURE 7. Comparison of the denoised images of the proposed method and the exiting method in one frame of video Tractor in the Set8 test set. The cyan framed area is enlarged for comparison. The results of noise reduction by Restormer, a lightweight state-of-the-art single-image denoising method, are also compared.

superior performance compared to the existing methods. In the non-blind case, the PTF Denoising Block receives a noise map representing the input image's noise level. Since this noise map assists the denoising network, the non-blind methods perform better. However, the blind case performs as well as or better than the non-blind method for the half model with only one PTF denoising block. This implies that the noise map-assisted network denoising is only valid for deeper models.

2) MODEL LIGHTNESS

Fig. 1 shows a bubble chart with the computational cost per 480p image (GMACs) on the horizontal axis, the PSNR value of the DAVIS test set at $\sigma = 50$ on the vertical axis, and the inference time per 480p image in Pytorch [17] as the bubble size. GMACs stands for Giga Multiply Accumulation Calculation. Multiply Accumulation Calculation is the number of multiplication and addition appearing in processing.

Tab. 3 and Fig. 1 show that PTFN can achieve high-quality denoising performance with tiny computational cost compared to existing methods. Specifically, PTFN outperforms BSVD with approximately 16.7% of its computational cost.

Both BSVD and PTFN have UNet-like structures, but while BSVD uses a classical ConvBlock, PTFN uses modern ConvBlock and PTF Block, which are lighter but have excellent performance. The ConvBlock used in BSVD is a two-layer Conv2d, and its computational cost is expressed by:

$$\underbrace{9hwc^2}_{3\times 3\ \text{Conv2d}} + \underbrace{9hwc^2}_{3\times 3\ \text{Conv2d}} = 18hwc^2 \tag{6}$$

The computational cost in PTFN's ConvBlock is expressed by:

$$\underbrace{2hwc^2}_{1\times 1\ \text{Conv2d}} + \underbrace{18hwc}_{3\times 3\ \text{DWConv2d}} + \underbrace{2hwc^2}_{1\times 1\ \text{Conv2d}} = 4hwc^2 + 18hwc \tag{7}$$

Note that h, w, and c are the feature map's height, width, and number of channels. When $h, w, c = 480, 856, 32$, the computational cost of the BSVD ConvBlock is 7.57×10^9,

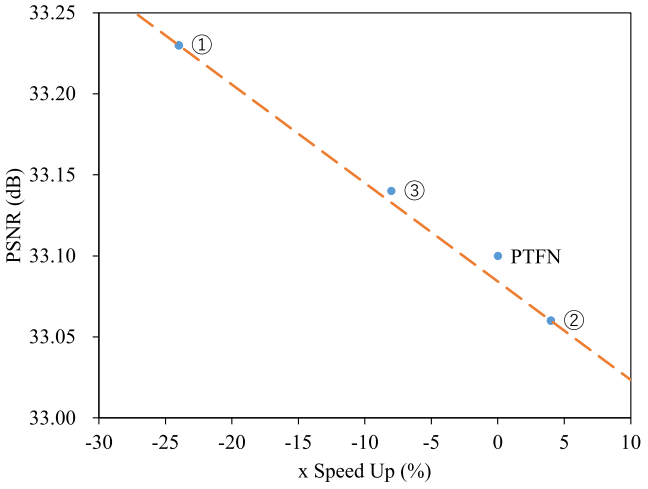

FIGURE 8. The horizontal axis is the change in the velocity versus PTFN for items other than ④ in Tab. 4, and the vertical axis is the value of PSNR for noise level $\sigma = 50$ in the DAVIS test set. The orange line is the straight line connecting ① and ②.

whereas the ConvBlock used in PTFN is 1.92×10^9, a significant reduction. PTFN Half shows superior performance with approximately 15.4% of FastDVDNet's computational cost, and the inference speed has also improved. The proposed method is also superior in terms of memory consumption. BSVD could not process 1080p videos on the NVIDIA RTX 3090 24GB, while PTFN can process 1080p videos. The lighter PTFN Half can process even higher resolution images with the same GPU. It can process 2K resolution (2560×1440) videos in 0.22 s per image.

D. QUALITATIVE COMPARISON

Fig. 6 and Fig. 7 compare the denoised images in one frame of the Snowboard and Tractor videos in the Set8 test set. As can be seen by focusing on the sky in Fig. 6, PTFN can denoise more cleanly than the existing methods, indicating that it has superior denoising performance.

The ability to capture temporal relationships is closely related to the noise removal performance of video. As shown in Fig. 7, Restormer [23], the lightweight state-of-the-art single image denoising method, cannot restore delicate patterns such as characters. This is because the characters are completely distorted in the noisy image of Fig. 7. It means that restoration is impossible from a single image in this case. Therefore, to restore characters, it is necessary to capture temporal relationships, and this capability is essential in video denoising. Due to these facts, we proposed the PTF to capture temporal relationships better. Our method can restore characters compared to previous methods, which means that our method successfully captures temporal relationships.

E. ABLATION STUDIES

Tab. 4 shows an ablation study of the PTFN network structure, verified in terms of denoising performance and inference speed.

1) STRUCTURE OF ConvBlock

Although [12], [13] proposed modern ConvBlocks with better performance than classical ConvBlocks, it is unclear whether they are suitable for video denoising. Therefore, in this paper, we also examine the ConvBlocks to find one suitable for video denoising.

① in Tab. 4 shows the case where the kernel size of the Depthwise Conv in the ConvBlock was changed from 3 to 7. In this case, the PSNR value for a noise level of $\sigma = 50$ in the DAVIS test set improved by 0.13 dB. A larger kernel size can capture more spatial redundancy in the image, resulting in improved performance. However, the processing time per 1080p image in PyTorch increased by 24%.

In Tab. 4, ② and ③ represent the case where the Depthwise Conv position of the ConvBlock is moved up. ② represents the case where the kernel size is 3, and ③ represents the case where the kernel size is 7. Moving up the Depthwise Conv position is introduced in [12], and its effectiveness in the image classification task has been demonstrated. The value of PSNR in ② is reduced by 0.04 dB, but the speed is also reduced by 4%, while the ③ value of PSNR is increased by 0.04 dB, but the speed is slowed down by 8%.

Fig. 8 verifies the trade-off between performance and speed for the items other than ④ in Tab. 4. From Fig. 8, It can be interpreted that PTFN, which has a kernel size of 3 and does not move up the position of the Depthwise Conv, is the most optimal model in terms of the trade-off between speed and performance.

2) PSEUDO TEMPORAL FUSION (PTF)

In order to verify the effectiveness of the PTF, a comparison between PTFN and ④ when the PTF of the PTF Block is replaced with GELU is conducted. Tab. 4 shows that the PSNR value of the DAVIS test set increased by 0.32 dB by introducing PTF, suggesting that the PTF improved the quality of the generated video by enhancing the ability to capture the temporal relationships of the network.

3) INTERMEDIATE LOSS

Tab. 5 compares the performance when the value of α in the loss function (5) is changed. The value of $\alpha = 0$ represents the case where Intermediate Loss is not used. Tab. 5 shows that the best performance is obtained when training with $\alpha = 0.1$. Since multi-step denoising is helpful for high-quality denoising, Intermediate Loss for multi-step denoising improves the network's performance.

4) SCALE OF PSEUDO TEMPORAL FUSION DENOISING BLOCK

There are three downsampling operators in Pseudo Temporal Fusion Denoising Block (PTF Denoising Block), with calculations on four different scales. Tab. 6 compares video denoising results for PTF Denoising Block scales of two and three. The depth of the model is adjusted to keep the processing speed of 1080p images in PyTorch constant.

TABLE 4. The results of ablation studies of the PTFN network structure. The PSNR values are calculated in the DAVIS test set with noise level $\sigma = 50$. The GPU used for the speed evaluation is an NVIDIA RTX 3090 24GB, and the framework used was PyTorch.

Model	Moved Up	Kernel Size	PSNR (dB)	Runtimes (s/1080p image)	× Speed Up
①	No	7	33.23	0.31	-24.00%
②	Yes	3	33.06	0.24	4.00%
③	Yes	7	33.14	0.27	-8.00%
④	No	3	32.78	0.21	16.00%
PTFN	No	3	33.10	0.25	0.00%

TABLE 5. PSNR values for the DAVIS test set when varying the coefficient of the Intermediate Loss term of the loss function. The best values are in bold.

α	PSNR (dB)
0.0	33.02
0.1	**33.10**
1.0	32.91

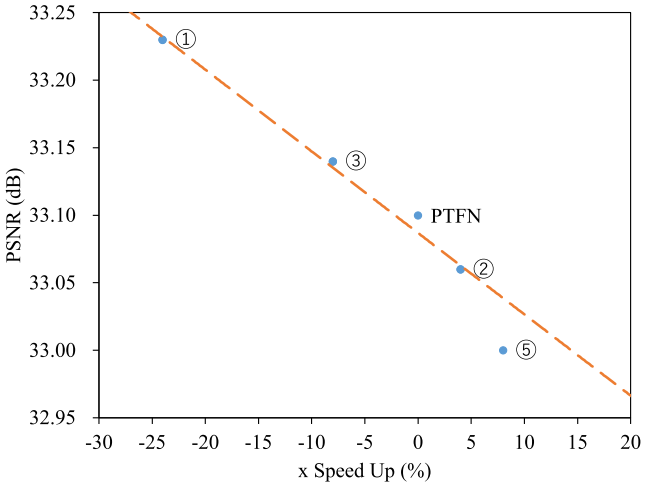

FIGURE 9. The horizontal axis is the change in the velocity versus PTFN for items other than ④ in Tab. 7, and the vertical axis is the value of PSNR for noise level $\sigma = 50$ in the DAVIS test set. The orange line in Fig. 9 is the approximate straight line for samples with PTF (①, ② ③ and PTFN).

Table 6 shows that a scale of four gives the best performance. Denoising at a variety of scales increases the virtual receptive field and improves performance.

V. CONCLUSION

In this paper, we propose the Pseudo Temporal Fusion Network (PTFN) for video denoising, which supports high performance and computational effeciency at the same time. PTFN are much less computationally expensive than existing video denoising methods, and their performance is significantly improved. PTFN is also superior in memory consumption and can process 1080p images on a GPU with 24GB RAM. PTFN employs a new Pseudo Temporal Fusion module, which captures pseudo-temporal relationships when combined with the Temporal Shift Module, contributing to performance. This paper also searches for a more suitable modern ConvBlock for video denoising. We demonstrated that PTFN is a valuable network for video denoising from various perspectives.

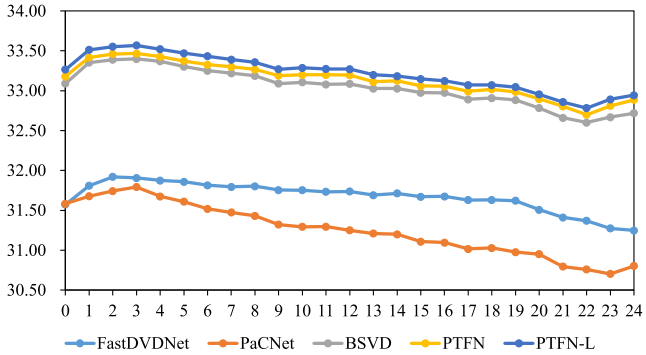

FIGURE 10. PSNR of each frame numbered 0 to 24 on snowboard of Set8 testset for the proposed and existing methods. The horizontal axis is the frame number and the vertical axis is the PSNR.

APPENDIX A
COMPARING PSEUDO TEMPORAL FUSION AND SIMPLEGATE

The Pseudo Temporal Fusion (PTF) proposed in this paper divides the feature map in the channel direction and then performs element-wise computation. This operation is similar to the SimpleGate (SG) proposed by Chen et al. [13]. However, there are clear differences between PTF and SG. We will discuss the difference between them from the two viewpoints of "Purpose" and "Effect on video denoising".

A. PURPOSE

PTF aims to efficiently capture the temporal relationships of video sequences in conjunction with the Temporal Shift Module (TSM) [16], which merges the temporal and channel relationships of feature maps. Therefore, PTF can capture pseudo-temporal relationships by operating feature maps in channel axis. Moreover, Since the TSM shifts the features one frame before and after, the PTF also divides the feature map into three equal parts. On the other hand, Chen et al. considered GELU as a special case of Gated Linear Units (GLU) and replaced them with SG, simpler GLU, to speed up the network. So for video denoising, PTF is more suitable than SG.

B. EFFECT ON VIDEO DENOISING

Tab. 7 is the comparison of PTFN and other networks with different settings. ① in Tab. 7 shows the case where the kernel size of the Depthwise Conv in the ConvBlock was changed from 3 to 7. ② and ③ represent the case where the Depthwise Conv position of the ConvBlock is moved up. ② represents the case where the kernel size is 3, and ③ represents the case

TABLE 6. Comparison of PTF Denoising Block at different scales. Num Scale means the number scales in the PTF Denoising Block. Depths at *n*th scale means the number of the processing block (TSM, ConvBlock, PTF Block) for the encoder and decoder at the scale where the PTF Denoising Block is downsampled *n* times. In the case of the bottom scale, a single value is shown because there is no distinction between the encoder and the decoder. The GPU used for the speed evaluation is an NVIDIA RTX 3090 24GB, and the framework used was PyTorch. The PSNR values are calculated in the DAVIS test set with noise level $\sigma = 50$. The best PSNR value for each item is bolded.

	Depths at nth scale				Runtimes	PSNR (dB)
Num Scale	$n = 0$	$n = 1$	$n = 2$	$n = 3$	(s/1080p image)	$\sigma = 50$
2	[1,1]	4	N/A	N/A	0.25	32.05
3	[1,1]	[1,1]	4	N/A	0.25	32.78
4	[1,1]	[1,1]	[1,1]	3	0.25	**33.10**

TABLE 7. The results of ablation studies of the PTFN network structure. The PSNR values are calculated in the DAVIS testset with noise level $\sigma = 50$. The GPU used for the speed evaluation is an NVIDIA RTX 3090 24GB, and the framework used was PyTorch.

Model	Moved Up	Module	Kernel Size	PSNR (dB)	Runtimes (s/1080p image)	× Speed Up
①	No	PTF	7	33.23	0.31	-24.00%
②	Yes	PTF	3	33.06	0.24	4.00%
③	Yes	PTF	7	33.14	0.27	-8.00%
④	No	GELU	3	32.78	0.21	16.00%
⑤	No	SimpleGate	3	33.00	0.23	8.00%
PTFN	No	PTF	3	33.10	0.25	0.00%

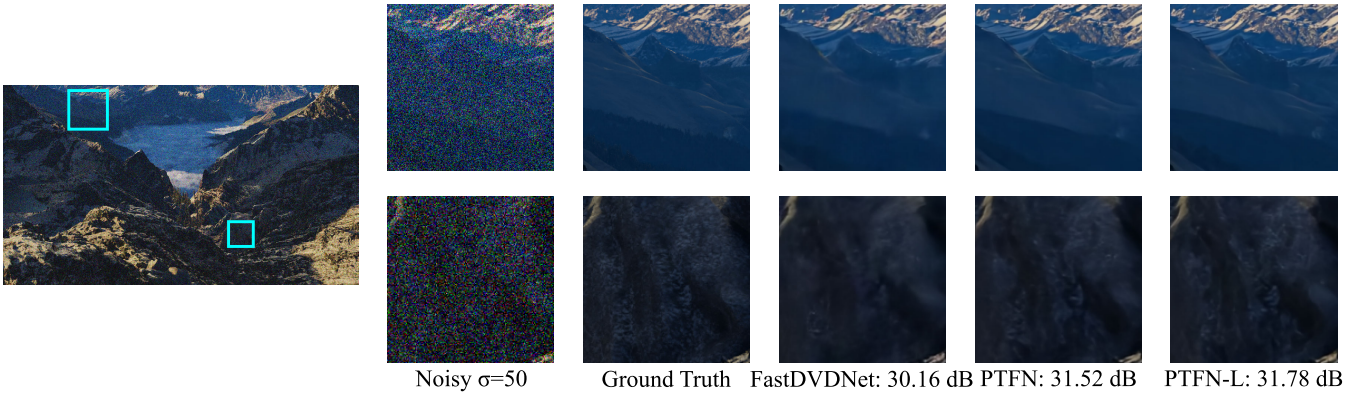

FIGURE 11. Comparison of the denoised images of the proposed method and the existing method in one frame of video 0001 in the spring dataset. The cyan framed area is enlarged for comparison. To the right of the method's name is the average of the PSNR in video 0001 in the spring dataset.

where the kernel size is 7. ④ shows the case when PTF is replaced with GELU. The case where the PTF is replaced by SimpleGate (⑤) is added from Tab. 4 in the main paper. Fig. 9 is a graph of Tab. 7.

Tab. 7 shows that PSNR is 0.10 dB higher when using PTF than when replacing it with SG. The orange line in Fig. 9 is the approximate straight line for samples with PTF. It represents the trade-off between computational cost and PSNR when PTF is used. In Fig. 9, point of ⑤ is located on the lower side of the orange line. This indicates that PTF is superior performance of SG in terms of the trade-off.

APPENDIX B
PSNR VALUES IN VIDEO SEQUENCES

Fig. 10 shows the PSNR for each frame from number 0 to 24 on the snowboard of the Set8 testset [18], [30]. It can be seen that our method (PTFN, PTFN-L) achieves overall better PSNR than the existing methods [1], [2], [6]. PTFN replaces the TSM with a Bidirectional Buffer Block [2] during inference. Thus it is free from inefficiencies in inference such as overlapping sliding windows.

APPENDIX C
DENOISING RESULTS IN HIGH-RESOLUTION VIDEO

To verify that the proposed methods perform well on high-resolution videos, we add Gaussian noise with $\sigma = 50$ to Video 0001 in the Spring Dataset [33] and denoise it with the proposed method and FastDVDNet. Spring Dataset is a high-resolution and high-detail dataset with a resolution 1080p (1920 × 1080). Fig. 11 shows one of the frames of the video denoising results. Fig. 11 shows that our method reconstructs hillside lines and rock textures more detail than FastDVDNet. The PSNR in our methods is also higher than FastDVDNet.

REFERENCES

[1] M. Tassano, J. Delon, and T. Veit, "FastDVDnet: Towards real-time deep video denoising without flow estimation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 1354–1363.

[2] C. Qi, J. Chen, X. Yang, and Q. Chen, "Real-time streaming video denoising with bidirectional buffers," in *Proc. 30th ACM Int. Conf. Multimedia*, Oct. 2022, pp. 2758–2766.

[3] P. Arias and J.-M. Morel, "Video denoising via empirical Bayesian estimation of space-time patches," *J. Math. Imag. Vis.*, vol. 60, no. 1, pp. 70–93, Jan. 2018.

[4] M. Maggioni, G. Boracchi, A. Foi, and K. Egiazarian, "Video denoising, deblocking, and enhancement through separable 4-D nonlocal spatiotemporal transforms," *IEEE Trans. Image Process.*, vol. 21, no. 9, pp. 3952–3966, Sep. 2012.
[5] M. Tassano, J. Delon, and T. Veit, "DVDNET: A fast network for deep video denoising," in *Proc. IEEE Int. Conf. Image Process. (ICIP)*, Sep. 2019, pp. 1805–1809.
[6] G. Vaksman, M. Elad, and P. Milanfar, "Patch craft: Video denoising by deep modeling and patch matching," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2021, pp. 2157–2166.
[7] L. Xiang, J. Zhou, J. Liu, Z. Wang, H. Huang, J. Hu, J. Han, Y. Guo, and G. Ding, "ReMoNet: Recurrent multi-output network for efficient video denoising," in *Proc. AAAI Conf. Artif. Intell.*, vol. 36, Jun. 2022, pp. 2786–2794.
[8] J. Liang, J. Cao, Y. Fan, K. Zhang, R. Ranjan, Y. Li, R. Timofte, and L. Van Gool, "VRT: A video restoration transformer," 2022, *arXiv:2201.12288*.
[9] J. Liang, Y. Fan, X. Xiang, R. Ranjan, E. Ilg, S. Green, J. Cao, K. Zhang, R. Timofte, and L. Van Gool, "Recurrent video restoration transformer with guided deformable attention," 2022, *arXiv:2206.02146*.
[10] 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 16 × 16 words: Transformers for image recognition at scale," in *Proc. Int. Conf. Learn. Represent.*, 2020, pp. 1–22.
[11] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo, "Swin transformer: Hierarchical vision transformer using shifted windows," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2021, pp. 10012–10022.
[12] Z. Liu, H. Mao, C.-Y. Wu, C. Feichtenhofer, T. Darrell, and S. Xie, "A ConvNet for the 2020s," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2022, pp. 11976–11986.
[13] L. Chen, X. Chu, X. Zhang, and J. Sun, "Simple baselines for image restoration," 2022, *arXiv:2204.04676*.
[14] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," *Commun. ACM*, vol. 60, no. 6, pp. 84–90, May 2017.
[15] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2014, *arXiv:1409.1556*.
[16] J. Lin, C. Gan, and S. Han, "TSM: Temporal shift module for efficient video understanding," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 7083–7093.
[17] A. Paszke et al., "PyTorch: An imperative style, high-performance deep learning library," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 32. Red Hook, NY, USA: Curran Associates, 2019, pp. 8024–8035. [Online]. Available: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf
[18] J. Pont-Tuset, F. Perazzi, S. Caelles, P. Arbeláez, A. Sorkine-Hornung, and L. Van Gool, "The 2017 Davis challenge on video object segmentation," 2017, *arXiv:1704.00675*.
[19] L. Jovanov, A. Pizurica, S. Schulte, P. Schelkens, A. Munteanu, E. Kerre, and W. Philips, "Combined wavelet-domain and motion-compensated video denoising based on video codec motion estimation methods," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 19, no. 3, pp. 417–421, Mar. 2009.
[20] A. Buades and J.-L. Lisani, "Enhancement of noisy and compressed videos by optical flow and non-local denoising," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 30, no. 7, pp. 1960–1974, Jul. 2020.
[21] W. Shen, M. Cheng, G. Lu, G. Zhai, L. Chen, M. S. Asif, and Z. Gao, "Spatial temporal video enhancement using alternating exposures," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 32, no. 8, pp. 4912–4926, Aug. 2022.
[22] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional networks for biomedical image segmentation," in *Proc. Int. Conf. Med. Image Comput. Comput.-Assist. Intervent.* Springer, 2015, pp. 234–241.
[23] S. W. Zamir, A. Arora, S. Khan, M. Hayat, F. S. Khan, and M. Yang, "Restormer: Efficient transformer for high-resolution image restoration," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2022, pp. 5728–5739.
[24] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 770–778. [Online]. Available: http://ieeexplore.ieee.org/document/7780459
[25] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He, "Aggregated residual transformations for deep neural networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 1492–1500.
[26] W. Shi, J. Caballero, F. Huszár, J. Totz, A. P. Aitken, R. Bishop, D. Rueckert, and Z. Wang, "Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 1874–1883.
[27] J. L. Ba, J. R. Kiros, and G. E. Hinton, "Layer normalization," 2016, *arXiv:1607.06450*.
[28] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 1251–1258.
[29] D. Hendrycks and K. Gimpel, "Gaussian error linear units (GELUs)," 2016, *arXiv:1606.08415*.
[30] Xiph Foundation. (2022). *Derf's Test Media Collection*. [Online]. Available: https://media.xiph.org/video/derf/
[31] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, *arXiv:1412.6980*.
[32] I. Loshchilov and F. Hutter, "SGDR: Stochastic gradient descent with warm restarts," 2016, *arXiv:1608.03983*.
[33] L. Mehl, J. Schmalfuss, A. Jahedi, Y. Nalivayko, and A. Bruhn, "Spring: A high-resolution high-detail dataset and benchmark for scene flow, optical flow and stereo," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2023, pp. 4981–4991.

KEI SHIBASAKI received the B.E. degree in electrical and information engineering from Keio University, Yokohama, Japan, in 2022, where he is currently pursuing the M.E. degree, under the supervision of Prof. Masaaki Ikehara. His research interests include image processing and deep learning.

MASAAKI IKEHARA (Senior Member, IEEE) received the B.E., M.E., and Ph.D. degrees in electrical engineering from Keio University, in 1984, 1986, and 1989, respectively. He is currently a Full Professor with the Department of Electronics and Electrical Engineering, Keio University. His research interests include multi-rate signal processing, wavelet image coding, and filter design problems.