I.J. Image, Graphics and Signal Processing, **2025, 3, 1-11**
Published Online on June 8, 2025 by MECS Press (http://www.mecs-press.org/)
DOI: 10.5815/ijigsp.2025.03.01

LiteDVDNet: Optimizing FastDVDNet for High-Speed Video Denoising

Andrii Ilchenko*
National Technical University of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute"
Department of Computer Engineering, Kyiv, 03056, Ukraine
E-mail: andreyilch87@gmail.com
ORCID ID: https://orcid.org/0009-0001-6781-3079
*Corresponding Author

Sergii Stirenko
National Technical University of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute"
Department of Computer Engineering, Kyiv, 03056, Ukraine
E-mail: stirenko@comsys.kpi.ua
ORCID ID: https://orcid.org/0000-0001-5478-0450

Received: 22 July, 2024; Revised: 04 March, 2025; Accepted: 24 March, 2025; Published: 08 June, 2025

Abstract: The growing demand for high-quality video processing in real-time applications demands efficient denoising techniques that can operate swiftly while maintaining visual fidelity. Conventional approaches often struggle to balance these competing requirements, especially when dealing with high-resolution video streams or resource-constrained environments. This study aims to develop methods for accelerating video data denoising using deep convolutional neural networks while maintaining acceptable output quality. We selected the popular FastDVDNet denoising network, which operates on a sliding window principle, as our baseline for comparison and a starting point for our research. This paper proposes several modifications of FastDVDNet that significantly enhance computational efficiency. We introduce four key optimizations: caching intermediate denoising results, reducing intermediate channels in input block, simplifying convolutional blocks, and halving the number of channels. We evaluated these modifications on the Set8 dataset and compared the results with the original model at various noise levels. Finally, we introduce LiteDVDNet, a fine-tuned version of FastDVDNet model that achieves the optimal balance between processing speed, and denoising performance. We developed two model variants: LiteDVDNet-32, which is 3× faster than the original model with only 0.18 dB average PSNR reduction, and the more lightweight LiteDVDNet-16, which delivers a 5× speed improvement at the cost of 0.61 dB average PSNR reduction.

Index Terms: Video Denoising, Efficient Inference, Deep Neural Networks, Deep Learning

1. Introduction

The worldwide COVID-19 crisis has triggered a massive shift to remote activities, leading to an extraordinary increase in the use of real-time video conferencing. Organizations, educational institutions, and households have widely adopted video chat platforms as primary communication tools. This trend continues to gain momentum as remote work and learning establish themselves as viable long-term strategies. However, user satisfaction in this new paradigm is tightly tied to the quality of video playback, which encompasses audio-video synchronization and media stream fidelity.

Despite advances in this domain, enhancing real-time video transmission quality remains a serious challenge. Video degradation stems from a multitude of sources, including hardware limitations (e.g., webcam defects and device processing constraints), environmental factors, network congestion, data integrity issues, and packet loss. The complexity of video denoising is further compounded by the sheer volume of data, the infeasibility of key frame retransmission, and the imperative of maintaining temporal coherence between denoised frames.

Furthermore, real-time video denoising demands frame-by-frame enhancement with minimal computational overhead, a requirement that challenges the limits of existing algorithms. This necessitates innovative approaches that can effectively balance the trade-offs between processing speed, resource utilization, and output quality. Addressing these challenges is crucial for improving the user experience and facilitating seamless communication in an increasingly digital world.

In this paper, we present LiteDVDNet, an enhanced iteration of the well-known FastDVDNet denoising model. Through a series of optimizations, this architecture considerably improves video data denoising speed, enabling real-time operations.

Our contribution can be summarized as follows:

- We introduce a frame-caching mechanism that exploits temporal redundancy in the denoising process, effectively reducing computational requirements by 50% with no quality degradation. Unlike the original approach that repeatedly processes the same frames, our method reuses already computed results.
- We demonstrate that the model input block can be significantly optimized by reducing intermediate channels count by a factor of three. Experimental results indicate that this modification yields a 24% speed improvement with only minimal quality reduction.
- We propose a simplified convolutional block design that achieves 16% faster computation with minimal quality impact, showing that redundant convolution operations can be eliminated without compromising the network's feature extraction capabilities.
- We present a parameter-efficient network variant with halved channel count, which reduces model size by 75% (from 2.48M to 0.64M parameters) and increases inference speed by almost 30% while maintaining acceptable denoising quality especially on lower noise levels, making it suitable for deployment on resource-constrained devices.

The combination of these results in the LiteDVDNet model, which achieves up to 5x inference speedup compared to the original FastDVDNet, while maintaining visual quality within acceptable parameters. The substantial reduction in computational overhead enables effective noise removal at higher resolutions, and represents a significant advancement for applications requiring real-time video processing, such as video conferencing and live streaming.

Source code is available at https://github.com/Merlin887/LiteDVDNet

2. Existing Approaches

In the context of video denoising algorithms, temporal consistency and, as result, elimination of flickering is a critical factor influencing the perceived quality of the output. To achieve these objectives, an algorithm must effectively utilize the temporal information present in adjacent frames when processing a specific frame within an image sequence.

Modern DNN approaches to the video data denoising can be roughly divided into three main categories: sliding window-based methods [1, 2], recurrent methods [3, 4, 5], and multiple input multiple output (MIMO) methods [5, 6].

Sliding Window Based Methods: These methods restore each frame using information from neighboring frames. The memory usage is directly proportional to the window size, but the results are coherent and the output video is smooth. However, this approach leads to redundant computations since each frame is fed into the network multiple times, slowing down the video stream processing, although it potentially allows for real-time operation. Architectures such as DVDNet [1] and FastDVDNet [2] fall under the category of sliding window methods.

Recurrent Methods: These methods utilize previously cleaned-up frames as a reference for restoring subsequent ones. They can be either unidirectional or bidirectional. Unidirectional techniques, suitable for streaming, rely only on past data, potentially limiting their noise-removal capability. In contrast, bidirectional techniques use information from both past and future frames, but this requires having the whole video available in advance, making real-time noise reduction impossible. Examples of recurrent networks include architectures such as BasicVSR [3], EMVD [4], and ReMoNet [5]. The latter also belongs to the MIMO category.

Multiple Input Multiple Output (MIMO) Structures: These structures process blocks of video frames in a single pass, using techniques such as channel shifting or local temporal windows. However, their memory consumption grows linearly with the number of frames processed, requiring long videos to be split into short clips. Like bidirectional recurrent methods, MIMO structures also experience performance degradation at the boundaries of these clips [8]. An example of such an architecture is VRT [6].

3. Original Method

Let's first give a brief description of the FastDVDNet baseline model presented in the original paper [2] before we proceed to the proposed improvements.

FastDVDNet is a cascaded two-step architecture designed for video denoising. When denoising a given frame at time **t**, the network takes five consecutive frames as input: $\{\tilde{I}_{t-2}, \tilde{I}_{t-1}, \tilde{I}_{t}, \tilde{I}_{t+1}, \tilde{I}_{t+2}\}$ The denoising is performed in 2 stages. On the first stage, three identical U-Net [7] blocks (Denoising Block 1) operate, which share the same weights. Each block takes a triplet of consecutive frames as input along with a noise map. On the second stage, a fourth U-Net block (Denoising Block 2) takes the outputs of the three blocks from the first stage as input. It has the same architecture as Denoising Block 1, but has different weights. The output of this block is the final denoised estimate of the central input frame $\tilde{I}_{t}$. The whole inference schema is shown on Fig.1.

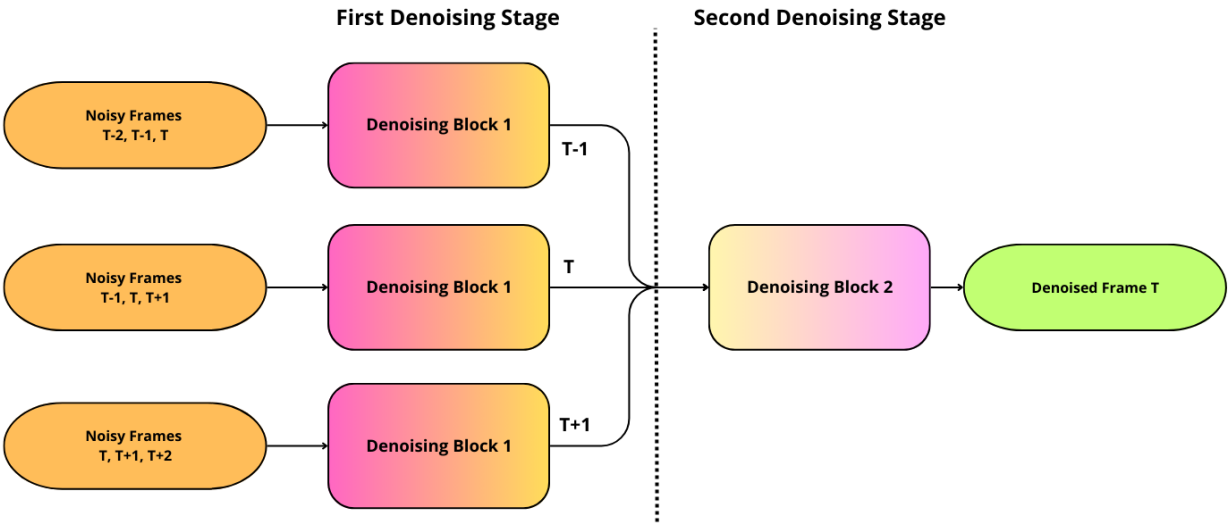

Fig. 1. FastDVDNet video denoising network

Both Denoising Block 1 and Denoising Block 2 share the same architecture, which is a modified multi-scale U-Net [7]. The network merges encoder and decoder features using pixel-wise addition instead of channel-wise concatenation to reduce memory requirements. It implements residual learning with a residual connection [10] between the central noisy input frame and the output. PixelShuffle block [9] is used for upsampling in the decoder to reduce gridding artifacts.

Most layers use point-wise ReLU activation functions, except for the last one. Batch normalization [11] is placed between convolutional and ReLU layers.

A crucial aspect of FastDVDNet is that it avoids explicit motion estimation/compensation. Instead, the capacity to handle motion is inherently embedded into the architecture through its multi-scale nature and cascaded design. This allows the network to implicitly deal with movement in the video sequences without a dedicated motion estimation step, contributing to its fast performance.

4. Proposed Improvements

Based on our analysis of FastDVDNet network architecture and extensive experimental validation, we identified four key areas for optimization.

Caching Results of First-Stage Block Computations: Video denoising involves significant temporal redundancy when processing consecutive frames with overlapping temporal windows. By analyzing the sliding window approach of FastDVDNet, we observed significant computational redundancy, as each frame is processed multiple times. We developed a caching mechanism that stores and reuses intermediate results from previous frames, effectively reducing the computational requirements by nearly 50%.

Reducing Intermediate Channels in Input Block: Deep networks often contain redundant feature channels that contribute little to overall performance. Studies in network pruning [20, 21] have demonstrated that many parameters in convolutional networks are effectively unused during inference. The number of intermediate channels in the InputCvBlock significantly impacts model performance, as it directly determines the number of convolution operations performed on the original image. By reducing the number of intermediate channels by a factor of three (from 90 to 30), we achieved substantial speedup with minimal quality loss.

Simplifying Convolutional Blocks: In deep networks, not all parameters contribute equally to model performance. The second convolution in each block frequently learns near-identity mappings or redundant features that contribute minimally to discriminative power [22]. Additionally, the U-Net architecture's skip connections already preserve critical spatial information [7], reducing the need for complex feature extraction at each level as demonstrated by Drozdzal et al. [23]. While removing a convolution does reduce the theoretical receptive field, the multi-scale nature of the network ensures sufficient contextual information is still captured [24]. By simplifying these blocks to use only a single convolution operation, we reduced computational requirements while maintaining strong feature extraction capabilities.

Halving Model Channels: Multiple studies have shown that neural networks can maintain performance with significantly fewer parameters than their original design [20, 22]. By halving the number of channels throughout our network, we achieved a four-fold reduction in model size while preserving acceptable denoising performance, particularly at lower noise levels.

The following parameters were introduced to the FastDVDNet framework to support our proposed modifications:

- **inference_mode**: Basic/Cached
- **interm_ch**: 90/30
- **simple_cv**: False/True
- **channels**: [32, 64, 128] / [16, 32, 64]

We then trained the neural network taking these parameters into account in order to test their impact on the neural network's performance and the quality of the results.

5. Training Details

The training dataset was constructed by corrupting clean image patches sourced from the DAVIS database [12] with additive white Gaussian noise (AWGN). While AWGN is a simplification of real-world noise, we use it to enable direct comparison with existing denoising methods in the literature that primarily employ this noise model as a standard benchmark.

Similar to the original FastDVDNet, our model accepts a sequence of noisy frames together with a noise map as input. This map provides the network with explicit spatial information about noise distribution, enabling it to potentially handle more complex and realistic noise patterns in practical applications.

The AWGN noise addition is mathematically represented as:

$$I_{noisy} = I_{clean} + N_{\sigma} \tag{1}$$

where I_{clean} is the clean image patch, N_{σ} is zero-mean white Gaussian noise with standard deviation σ, and I_{noisy} is the resulting noisy patch.

Each training sample consists of 5 spatial patches cropped at the same location in contiguous frames, taken from randomly sampled sequences of the training dataset. Noisy frames are generated by adding AWGN of $\sigma \in [5, 55]$ to clean patches of a given sequence. Noise map is created with all elements equal to the σ value used for that particular sample.

To improve model generalization, we augment our training data with rescaling and random flips using the NVIDIA DALI Loader library.

The loss function is the L2 norm between the network output and the clean central patch.

A total of 32,000 training samples are used for each epoch, with a spatial patch size of 96x96. The model is implemented in PyTorch [13] and optimized using the ADAM algorithm [14] with default hyperparameters. Training runs for 80 epochs with a mini-batch size of 64. A step decay learning rate schedule is employed, starting at 1e-3 for the first 10 epochs, then reducing by half every 10 epochs.

Also, convolutional kernel orthogonalization is applied during the first 60 epochs for regularization, which has been shown to benefit network performance [15].

6. Evaluation and Metrics

For our network evaluation, we employ several key metrics to assess the performance of our changes. These metrics include Peak Signal-to-Noise Ratio (PSNR), Structural Similarity Index (SSIM) [17], Spatio-Temporal Reduced Reference Entropic Differences (ST-RRED) [16], and the mean computation time for the denoised frame inference.

PSNR serves as our primary quantitative metric due to its widespread use in denoising literature, allowing direct comparisons with existing approaches. However, as PSNR does not fully capture perceptual quality, we complement it with SSIM which better correlates with human visual perception by evaluating structural similarities between frames. For video-specific quality assessment, ST-RRED is particularly valuable as it quantifies both spatial and temporal distortions, making it especially suitable for evaluating video denoising methods where temporal consistency is crucial**.**

Additionally, by measuring the computation time for each frame, we assess the efficiency of our algorithm, an important factor for real-time processing scenarios. For implementation consistency, both ST-RRED and SSIM scores are calculated using the ***scikit-video*** library.

The testing is conducted on the SET8 dataset, which allows us to benchmark our algorithm against a diverse range of videos. Since the video recordings in the SET8 dataset vary in length, the evaluation was conducted using the first 30 frames of each recording. To simulate diverse noise conditions, we introduce Additive White Gaussian Noise (AWGN) at various levels, ranging from 10 to 50. This wide range of noise intensities helps to understand how our network performs under different degrees of image degradation.

The proposed architecture was evaluated on a workstation equipped with an Intel Core i5-10600K processor, an NVIDIA RTX 3060 GPU with 12GB VRAM, and 32GB of system RAM.

torch.backends.cudnn.deterministic is set to **True** during tests to receive clean results, unaffected by CUDNN runtime optimization.

7. Implementation and Results

Let's go through each of the proposed improvements in more detail and demonstrate the results of their practical application.

7.1. Caching Results of First-Stage Blocks Computations

To understand why caching is effective, let's consider how the original FastDVDNet processes consecutive frames. When denoising frame N, the network processes five consecutive noisy frames: $\{\tilde{I}_{N-2}, \tilde{I}_{N-1}, \tilde{I}_N, \tilde{I}_{N+1}, \tilde{I}_{N+2}\}$. These frames are processed through three first-stage denoising blocks, which generate three intermediate outputs $\{I_{N-1}, I_N, I_{N+1}\}$.

When moving to denoise frame N+1, the network processes $\{\tilde{I}_{N-1}, \tilde{I}_N, \tilde{I}_{N+1}, \tilde{I}_{N+2}, \tilde{I}_{N+3}\}$, generating new outputs $\{I_N, I_{N+1}, I_{N+2}\}$. Notice that two of these three frames $\{I_N, I_{N+1}\}$ were already created in the previous step.

Our caching strategy exploits this temporal redundancy by:

- **Storing intermediate results**: We maintain a cache of the outputs from the first denoising stage for the previous frame.
- **Selective recomputation**: Instead of recomputing all three first-stage outputs for each new frame, we reuse two cached outputs and compute only one new output.
- **Cache updating**: After each frame is processed, we update the cache by removing the oldest entry and adding the newly computed result.

Fig. 2. illustrates this process:

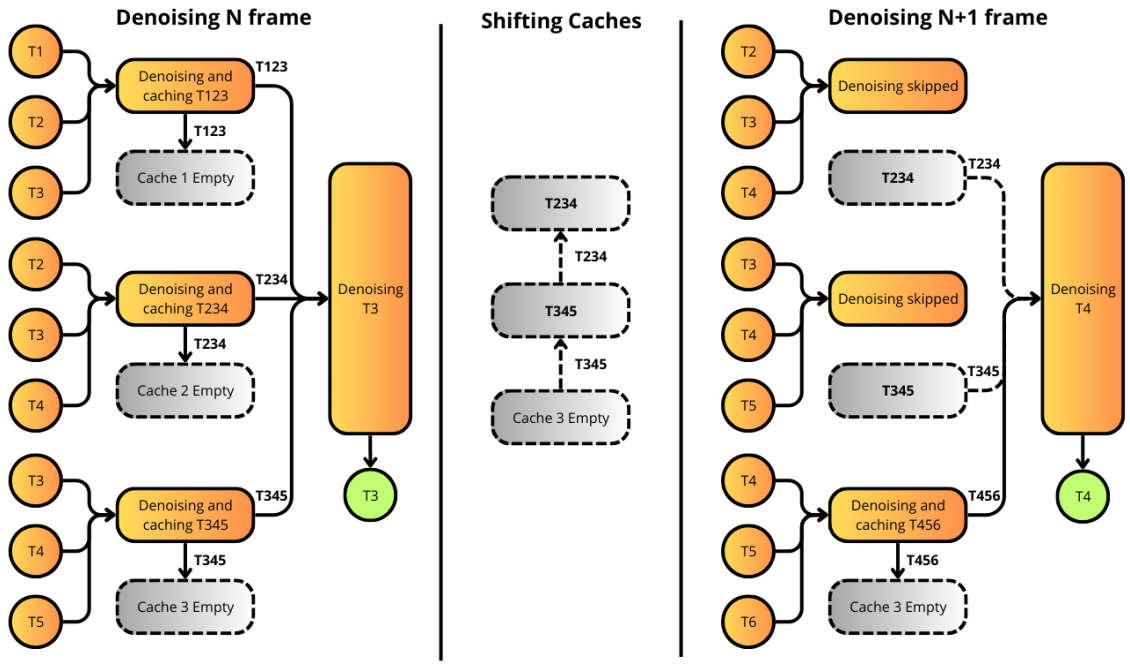

Fig. 2. Cached inference schema

To empirically validate our rationale, we evaluate this approach and compare the relevant metrics against the baseline model. Table 1 presents the testing results of FastDVDNet with caching enabled.

Table 1. Cached inference performance comparison with baseline model. For PSNR and ST-RRED: larger is better; SSIM and Denoise Time: smaller is better.

Model	Noise Level	PSNR	ST-RRED	SSIM	Denoise Time (msec)
Baseline / Cached	10	35.50 / 35.50	0.98 / 0.98	2.85 / 2.99	81 / **41**
Baseline / Cached	20	32.66 / 32.66	0.96 / 0.96	7.12 / 7.06	81 / **41**
Baseline / Cached	30	30.91 / 30.91	0.94 / 0.94	13.58 / 13.28	81 / **41**
Baseline / Cached	40	29.68 / 29.68	0.92 / 0.92	21.86 / 22.41	81 / **41**
Baseline / Cached	50	28.74 / 28.74	0.90 / 0.90	32.16 / 32.74	81 / **41**

The results demonstrate that the quality remains identical to the baseline, while the inference speed improves nearly twofold. The minor discrepancy in SSIM is related to computational details of this metric in the ***scikit-video*** library. Moreover, an advantage of this caching strategy is that it does not require retraining of the already trained network. Also, it is worth noting that cached inference mode is suitable only for computations, not for network training.

7.2. Reduce InputCvBlock intermediate channels by a factor of 3

In the original implementation of InputCvBlock (see left side in Fig.3), the number of intermediate channels in the first convolution is set to 90.

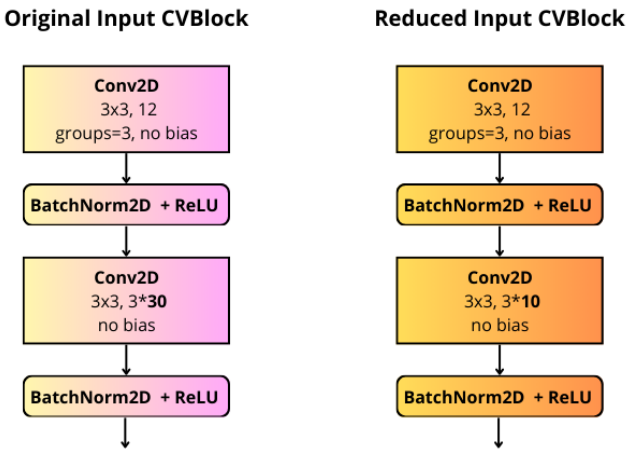

Fig. 3. Original and reduced InputCvBlock comparison

We propose reducing the number of such channels by a factor of three to speed up the frame denoising process. Table 2 presents the test results of the model trained with a reduced number of channels.

Table 2. Reduced InputCvBlock performance comparison with baseline model. For PSNR and ST-RRED: larger is better; SSIM and Denoise Time: smaller is better

Model	Noise Level	PSNR	ST-RRED	SSIM	Denoise Time (msec)
Baseline / Reduced	10	**35.50** / 35.48	0.98 / 0.98	2.92 / 2.92	81 / **65**
Baseline / Reduced	20	**32.66** / 32.53	0.96 / 0.96	**7.16** / 8.11	81 / **65**
Baseline / Reduced	30	**30.91** / 30.76	0.94 / 0.94	**13.55** / 16.89	81 / **65**
Baseline / Reduced	40	**29.68** / 29.53	**0.92** / 0.91	**21.87** / 28.76	81 / **65**
Baseline / Reduced	50	**28.74** / 28.59	**0.90** / 0.89	**32.65** / 43.96	81 / **65**

As a result of testing, we observe an average speed gain of 16 msec, which constitutes an approximate 24% improvement. The cost of this acceleration is a decrease in PSNR by an average of 0.12 dB, which is generally an attractive trade-off. However, it's important to note that the quality degradation follows a pattern correlated with noise level. At σ=10, the degradation is minimal (0.02 dB), while at σ=50, it increases to 0.15 dB, showing a more pronounced effect at higher noise levels.

The SSIM values also demonstrate this trend, with the difference between the baseline and reduced channel model growing from negligible at σ=10 to a more significant gap at σ=50 (32.65 vs 43.96). This pattern suggests that higher noise scenarios require more feature channels to effectively separate noise from signal, and reducing intermediate channels disproportionately affects the model's ability to handle severe noise. For applications where video streams contain high noise levels ($\sigma \geq 40$), this trade-off should be carefully considered against the speed benefits.

7.3. Using simplified convolutional blocks

The original convolutional block used in the DownBlock and UpBlock of FastDVDNet is shown on the left side of Fig.4.

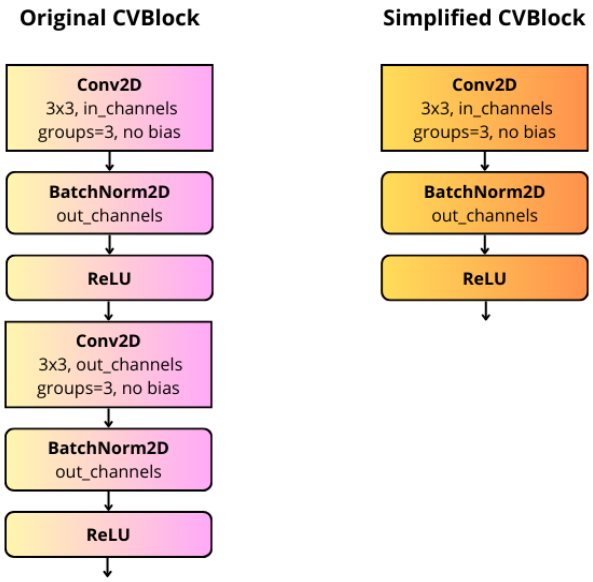

Fig. 4. Original and simplified convolution block comparison

In our implementation, we simplify this block by limiting it to only one convolutional operation, which is equivalent to Conv2d => BN => ReLU.

The test results for this modification can be found in Table 3.

Table 3. LiteCvBlock usage performance comparison with baseline model. For PSNR and ST-RRED: larger is better; SSIM and Denoise Time: smaller is better.

Model	Noise Level	PSNR	ST-RRED	SSIM	Denoise Time (msec)
Baseline / LiteCvBlock	10	**35.50** / 35.43	0.97 / **0.98**	2.95 / **2.86**	81 / **70**
Baseline / LiteCvBlock	20	**32.66** / 32.57	0.96 / 0.96	**7.13** / 7.40	81 / **70**
Baseline / LiteCvBlock	30	**30.91** / 30.82	0.94 / 0.94	**14.64** / 14.20	81 / **70**
Baseline / LiteCvBlock	40	**29.68** / 29.59	**0.92** / 0.91	**22.16** / 23.81	81 / **70**
Baseline / LiteCvBlock	50	**28.74** / 28.65	0.90 / 0.90	**32.06** / 35.21	81 / **70**

As a result, we observe a 16% computation speedup, with an average PSNR loss of 0.08, while the difference in characteristics such as ST-RRED and SSIM is practically negligible even at high noise levels. The quality degradation is remarkably consistent across different noise levels, ranging from 0.07 dB at σ=10 to 0.09 dB at σ=50, suggesting that this simplification affects the model's fundamental feature extraction capabilities rather than specifically its noise handling.

The ST-RRED metric shows negligible change at lower noise levels, indicating that temporal consistency is largely preserved by the simplified block. However, the SSIM metric does show a more pronounced degradation at higher noise levels (35.21 vs 32.06 at σ=50), suggesting some loss in structural information preservation when dealing with severe noise.

Overall, this modification offers one of the best speed-quality trade-offs among our optimizations, as it achieves a significant speed improvement with minimal and predictable quality impact across the entire noise range.

7.4. Halving model channels

FastDVDNet denoising block structure is described in Fig.5.

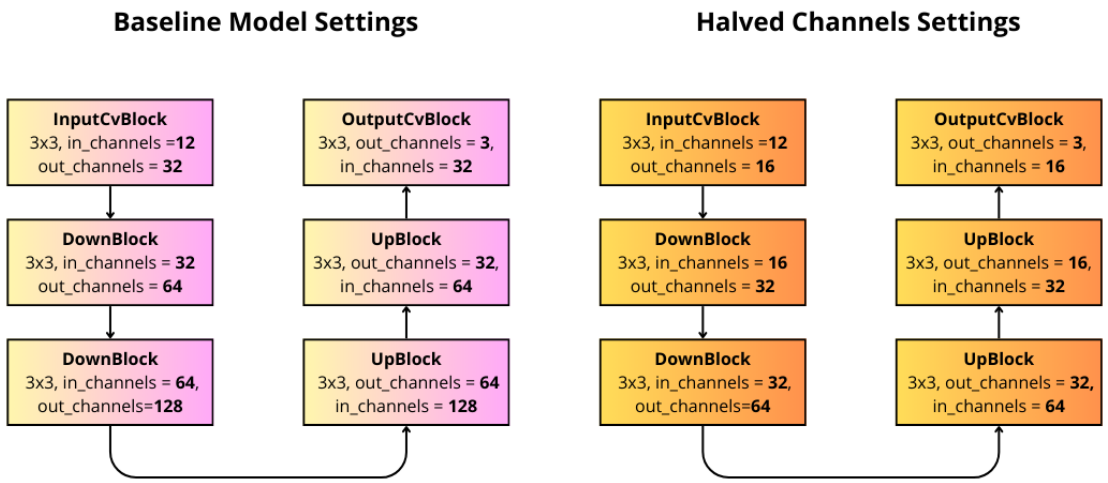

Fig. 5. FastDVDNet denoising block structure

In the baseline implementation, the number of model channels is set to *chs_lyr0=32, chs_lyr1=64, chs_lyr2=128*. We propose to halve the number of channels, by using *chs_lyr0=16, chs_lyr1=32, chs_lyr2=64*.

This modification reduces the number of model parameters by approximately 4 times - from 2,479,096 parameters in the original model to only 639,320 in the halved version.

The number of channels in convolutional layers directly affects both the model's computational requirements and size. To understand why halving the number of channels is so effective at reducing computational load, let's analyze the computational complexity of convolutional operations. For a convolutional layer with C_{in} input channels, C_{out} output channels, kernel size K, and feature map size H×W, the computational complexity is O ($C_{in} \times C_{out} \times K^2 \times H \times W$). When we decrease the channel count by half, the complexity is reduced to O (($C_{in}/2$) × ($C_{out}/2$) × K^2 × H × W).

Theoretically, this represents a 75% reduction in computational requirements. However, in practice, the actual speedup is less significant. The reason behind this is modern GPUs with CUDA cores are highly optimized for parallel operations, and PyTorch leverages these capabilities through libraries like cuDNN [25]. The PyTorch implementation of Conv2D utilizes several hardware-specific optimizations [26], memory access pattern optimization, and fusion of operations where possible [27]. Additionally, fixed overheads such as memory transfers between CPU and GPU, as well as the non-convolutional operations in the network, remain unchanged regardless of channel count.

Therefore, the effectiveness of such reduction should be experimentally verified. The results of this experiment are presented in Table 4.

Table 4. Halved channels performance comparison with baseline model. For PSNR and ST-RRED: larger is better; SSIM and Denoise Time: smaller is better.

Model	Noise Level	PSNR	ST-RRED	SSIM	Denoise Time (msec)
Baseline / Halved Channels	10	**35.50** / 35.25	0.98 / 0.98	**2.84** / 3.00	81 / **63**
Baseline / Halved Channels	20	**32.66** / 32.33	0.96 / 0.96	**7.15** / 8.66	81 / **63**
Baseline / Halved Channels	30	**30.91** / 30.56	**0.94** / 0.93	**13.37** / 17.89	81 / **63**
Baseline / Halved Channels	40	**29.68** / 29.33	**0.92** / 0.91	**21.84** / 30.65	81 / **63**
Baseline / Halved Channels	50	**28.75** / 28.38	**0.90** / 0.88	**33.75** / 46.79	81 / **63**

The computation time is thereby reduced by 29% with PSNR losses 0.33 dB average. This relatively modest degradation despite the three-quarter reduction in parameters (75%) suggests that the original model was overparameterized for the video denoising task, particularly at lower noise levels.

The quality degradation pattern clearly correlates with noise levels, starting at 0.25 dB for σ=10 and increasing to 0.37 dB for σ=50. This pattern is even more evident in the SSIM metric, where the difference grows from minimal at low noise levels to substantial at high noise levels (46.79 vs 33.75 at σ=50, representing a 38.6% increase).

The ST-RRED metric also shows progressive degradation as noise increases, dropping from identical performance at σ=10 to a measurable difference at σ=50 (0.90 vs 0.88). This suggests that temporal consistency, critical for video quality perception, is increasingly compromised at higher noise levels when using the halved-channel model.

These results indicate that for applications where videos contain predominantly low noise (σ≤20), the halved-channel model offers a viable trade-off, providing substantial size and speed benefits with acceptable quality loss. However, for applications dealing with high-noise scenarios (σ≥30), this optimization might compromise quality beyond acceptable thresholds.

7.5. Putting It All Together: LiteDVDNet

To create LiteDVDNet, we combined the optimization techniques described in the previous sections. This neural network utilizes cached output, reduces the InputCvBlock intermediate channels to 30, and employs a simplified version of CvBlock. We trained two versions of this model: one with the original number of channels (LiteDVDNet-32), and another with half the channels (LiteDVDNet-16). These optimizations transform the baseline FastDVDNet architecture into a more efficient model while preserving acceptable denoising quality.

To establish the effectiveness of LiteDVDNet in a broader context, we compared our proposed architecture's performance with other state-of-the-art video denoising methods on the Set8 dataset. For this purpose, we trained both versions of LiteDVDNet for an extended period to make their quality directly comparable with other solutions. Specifically, we used 256,000 training samples per epoch while maintaining all other training parameters unchanged.

Performance data for competing methods was obtained from the BSVD paper [8]. To ensure a fair comparison, we applied a normalization based on FastDVDNet runtime performance: we calculated the ratio between FastDVDNet's speed on our test server with RTX 3060 and its speed on the RTX 3090 server measured in [8]. This difference was approximately 2 times, and we used this factor to normalize all runtime measurements. Table 5 presents a comparison of LiteDVDNet variants against other denoising models.

Table 5. Comparison of LiteDVDNet variants with other video denoising methods of PSNR (dB), parameters and runtime on Set8 dataset. Runtime estimates for 960x540 resolution video normalized to our RTX 3060 GPU measurements. C and G represent CPU and GPU time cost, respectively.

Model	Runtime(s)	Params	PSNR σ=10	PSNR σ=20	PSNR σ=30	PSNR σ=40	PSNR σ=50	Average
PaCNet [19]	47.5 (G)	2.87M	**37.06**	**33.94**	32.05	30.70	29.66	32.68
VNLnet [18]	2.208 (G)	-	37.10	33.88	-	30.55	29.47	-
BSVD [8]	0.068 (G)	-	36.74	33.83	**32.14**	**30.97**	**30.06**	**32.75**
DVDnet [1]	9.462 (C+G)	1.3M	36.08	33.49	31.79	30.55	29.56	32.29
FastDVDnet [2]	0.081 (G)	2.48M	36.44	33.43	31.68	30.46	29.53	32.31
LiteDVDNet-32	0.027 (G)	1.7M	36.23	33.26	31.51	30.28	29.34	32.13
LiteDVDNet-16	**0.016 (G)**	**0.64M**	35.80	32.87	31.10	29.85	28.90	31.70

LiteDVDNet-32 achieves a 3× speedup compared to the original FastDVDNet with only a 0.18 dB reduction in average PSNR (32.13 dB vs 32.31 dB). This represents an excellent balance between computational efficiency and denoising performance. The more aggressive LiteDVDNet-16 pushes this trade-off further with a remarkable 5× speedup at the cost of a 0.61 dB reduction in average PSNR.

LiteDVDNet-16 requires only 0.64M parameters (a 75% reduction from FastDVDNet's 2.48M), yet maintains reasonable denoising quality despite the significant reduction in model complexity.

While high-quality models like PaCNet and BSVD achieve better denoising results than LiteDVDNet-32 (by 0.55-0.62 dB on average), they come with substantial computational costs. PaCNet is around 150 times slower, making it impractical for real-time applications. Even the most efficient BSVD is still 2.5× slower than LiteDVDNet-32 (0.068s vs 0.027s). At 68ms per frame, BSVD cannot achieve the 30 FPS threshold (33.3ms per frame) required for real-time video processing without expensive high-end hardware.

With processing times of 27ms and 16ms per frame for 960x540 resolution video respectively for LiteDVDNet-32 and LiteDVDNet-16, both models exceed the threshold for real-time processing at 30 FPS on a popular RTX 3060 GPU.

In video conferencing applications, LiteDVDNet can be integrated as a preprocessing filter after frame acquisition but before encoding, which places denoising earlier in the pipeline before compression artifacts are introduced. The model can be further enhanced with adaptive noise estimation, where a lightweight estimator analyzes each frame in real-time, allowing the network to adjust to changing lighting conditions.

Our optimizations are particularly valuable for notebooks, where the reduced computational requirements and smaller memory footprint improve both processing speed and energy efficiency.

Beyond video conferencing, LiteDVDNet is suitable for other live video applications including surveillance and streaming platforms. The significant speed improvements in LiteDVDNet-16 even allow for processing higher resolution content in real-time, addressing the growing demand for high-definition video denoising.

8. Conclusion and Future Research Directions

In this paper, we introduced LiteDVDNet, an optimized version of FastDVDNet that achieves up to 5× faster inference while maintaining acceptable denoising quality.

This level of performance enables real-time denoising even at resolutions exceeding 960x540, for which testing was conducted. This brings this type of network for denoising into the category of state-of-the-art methods in terms of execution speed while preserving decent video quality results.

Our contributions include an intermediate result caching mechanism that doubles inference speed of FastDVDNet without quality loss, alongside architectural optimizations that effectively balance speed and quality.

A promising direction for future work is a hybrid architecture that combines LiteDVDNet-32 denoising blocks in the first stage with the LiteDVDNet-16 block in the second stage. Since the first stage substantially reduces noise levels, the reduced denoising block becomes more effective for the second stage. Our preliminary analysis suggests this hybrid approach would process frames at a speed that is the arithmetic mean between both models while maintaining quality comparable to LiteDVDNet-32.

Additionally, further acceleration could be achieved through quantization and pruning techniques [28]. These approaches, when combined with our architectural improvements, have the potential to unlock even greater efficiency gains, making video denoising feasible on resource-constrained devices.

References

[1] Matias Tassano, Julie Delon, and Thomas Veit. 2019. DVDNet: A fast network for deep video denoising. In ICIP

[2] Matias Tassano, Julie Delon, and Thomas Veit. 2020. FastDVDNet: Towards real-time deep video denoising without flow estimation. In CVPR

[3] Kelvin C.K. Chan, Xintao Wang, Ke Yu, Chao Dong, and Chen Change Loy. 2021. BasicVSR: The Search for Essential Components in Video Super-Resolution and Beyond. In CVPR.

[4] Matteo Maggioni, Yibin Huang, Cheng Li, Shuai Xiao, Zhongqian Fu, and Fenglong Song. 2021. Efficient Multi-Stage Video Denoising with Recurrent Spatio Temporal Fusion. In CVPR.

[5] Liuyu Xiang, Jundong Zhou, Jirui Liu, Zerun Wang, Haidong Huang, Jie Hu, Jungong Han, Yuchen Guo, and Guiguang Ding. 2022. ReMoNet: Recurrent Multi-output Network for Efficient Video Denoising. In AAAI.

[6] Jingyun Liang, Jiezhang Cao, Yuchen Fan, Kai Zhang, Rakesh Ranjan, Yawei Li, Radu Timofte, and Luc Van Gool. 2022. VRT: A Video Restoration Transformer. arXiv:2201.12288 (2022).

[7] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. 2015. U-Net: Convolutional Networks for Biomedical Image Segmentation. In MICCAI.

[8] Chenyang Qi, Junming Chen, Xin Yang, Qifeng Chen. 2022 Real-time Streaming Video Denoising with Bidirectional Buffers, In ACM Multimedia

[9] Wenzhe Shi, Jose Caballero, Ferenc Huszar, Johannes Totz, Andrew P. Aitken, Rob Bishop, Daniel Rueckert, and Zehan Wang. 2016. Real-Time Single Image and Video Super-Resolution Using an Efficient Sub-Pixel Convolutional Neural Network. In IEEE

[10] Matias Tassano, Julie Delon, and Thomas Veit. An analysis and implementation of the FFDnet image denoising method. Image Processing On Line, 9:1–25, Jan 2019.

[11] Sergey Ioffe and Christian Szegedy. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In International Conference on Machine Learning (ICML), pages 448–456. JMLR.org, 2015.

[12] Anna Khoreva, Anna Rohrbach, and Bernt Schiele. Video object segmentation with language referring expressions. In ACCV, 2018.

[13] Adam Paszke, Gregory Chanan, Zeming Lin, Sam Gross, Edward Yang, Luca Antiga, and Zachary Devito. Automatic differentiation in PyTorch. 2017. In NIPS

[14] D.P. Kingma and J.L. Ba. ADAM: A Method for Stochastic Optimization. Proc. ICLR, 2015.

[15] Kai Zhang, Wangmeng Zuo, and Lei Zhang. FFDNet: Toward a fast and flexible solution for CNN-based image denoising. IEEE Transactions on Image Processing, Sep 2018.

[16] Rajiv Soundararajan and Alan C. Bovik. Video quality assessment by reduced reference spatio-temporal entropic differencing. IEEE Transactions on Circuits and Systems for Video Technology, 2013

[17] Zhou Wang, Alan C. Bovik, Hamid R. Sheikh, and Eero P. Simoncelli. 2004. Image quality assessment: from error visibility to structural similarity. IEEE Transactions on Image Processing, 13(4), 600-612

[18] Axel Davy, Thibaud Ehret, Jean Michel Morel, Pablo Arias, and Gabriele Facciolo. 2018. Non-Local Video Denoising by CNN. arXiv abs/1811.12758 (2018).

[19] Gregory Vaksman, Michael Elad, and Peyman Milanfar. 2021. Patch Craft: Video Denoising by Deep Modeling and Patch Matching. In ICCV.

[20] Song Han, Huizi Mao, and William J. Dally. 2016. Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding. International Conference on Learning Representations (ICLR).

[21] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the Knowledge in a Neural Network. arXiv preprint arXiv:1503.02531.

[22] Zhuang Liu, Jianguo Li, Zhiqiang Shen, Gao Huang, Shoumeng Yan, and Changshui Zhang. 2017. Learning Efficient Convolutional Networks through Network Slimming. In IEEE International Conference on Computer Vision (ICCV).

[23] Michal Drozdzal, Eugene Vorontsov, Gabriel Chartrand, Samuel Kadoury, and Chris Pal. 2016. The Importance of Skip Connections in Biomedical Image Segmentation. In Deep Learning and Data Labeling for Medical Applications.

[24] Dumitru Erhan, Christian Szegedy, Alexander Toshev, and Dragomir Anguelov. 2014. Scalable Object Detection Using Deep Neural Networks. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR).

[25] Sergei Chetlur, Cliff Woolley, Philippe Vandermersch, Jonathan Cohen, John Tran, Bryan Catanzaro, and Evan Shelhamer. 2014. cuDNN: Efficient Primitives for Deep Learning. arXiv preprint arXiv:1410.0759

[26] Andrew Lavin and Scott Gray. 2016. Fast Algorithms for Convolutional Neural Networks. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR)

[27] Tal Ben-Nun and Torsten Hoefler. 2019. Demystifying Parallel and Distributed Deep Learning: An In-Depth Concurrency Analysis. ACM Computing Surveys, 52(4)
[28] Liang, T., Glossner, J., Wang, L., Shi, S., & Zhang, X. (2021). Pruning and quantization for deep neural network acceleration: A survey. Neurocomputing, 461, 370-390.

Authors' Profiles

Andrii Ilchenko received a master's degree from the Department of Computing Engineering, National Technical University of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute". He is currently a Ph.D. student at the same university. His research interests include computer vision, deep neural networks, video processing, vision transformers, and other related areas.

Sergii Stirenko, Head of Computer Engineering Department, Research Supervisor of KPI-Samsung R&D Lab, Head of NVIDIA GPU Education and NVIDIA GPU Research Center, and Professor at National Technical University of Ukraine "Kyiv Polytechnic Institute." Research is mainly focused on artificial intelligence, high-performance computing, cloud computing, distributed computing, parallel computing, eHealth, simulations, and statistical methods. And published more than 60 papers in peer-reviewed international journals.

How to cite this paper: Andrii Ilchenko, Sergii Stirenko, "LiteDVDNet: Optimizing FastDVDNet for High-Speed Video Denoising", International Journal of Image, Graphics and Signal Processing(IJIGSP), Vol.17, No.3, pp. 1-11, 2025. DOI:10.5815/ijigsp.2025.03.01