Supplementary Material for ReBotNet: Fast Real-time Video Enhancement

Jeya Maria Jose Valanarasu[1,3*] Rahul Garg[1] Andeep Toor[1] Xin Tong[1] Weijuan Xi[1]
Andreas Lugmayr[2] Vishal M. Patel[3] Anne Menini[1]
[1]Google [2]ETH Zurich [3]Johns Hopkins University

1. Video Qualitative results

We direct the readers to this link to see qualitative comparisons in terms of videos of ReBotNet with other methods. In Figures 1 and 2, we provide more qualitative results on PortraitVideo and FullVideo datasets respectively.

2. Experiments leading to our design choice:

We experimented with ConvNets, ConvNexts, Transformers, and Mixers individually before finalizing our design choice—ReBotNet. These experiments yielded analytical insights crucial in shaping our decision." FastDVDNet and BasicVSR++ represent purely convolutional methods, whereas VRT and RVRT utilize transformer-based approaches with transposed convolution layers in their decoders. Despite their unique design improvements aimed at enhancing performance, these methods share a common encoder-decoder architecture, typical of most restoration techniques. To comprehend the advantages of encoder design, we conduct an experimental setup altering the encoder design while maintaining a fixed decoder with simple transposed convolution layers. Our objective is to match the performance of these models, aiming to gauge the computational requirements needed to achieve equivalent performance among these diverse encoders. The performance in terms of PSNR on the Portrait Video dataset and the latency of each of these models in terms of ms is reported in the table 14 below:

Table 1. Experiments with different encoders on PortraitVideo dataset.

Encoder	Latency (in ms)	PSNR
ConvNet	86.89	31.24
ConvNext	30.56	31.22
Transformer	140.58	31.25
Mixer	28.65	31.23

We can observe that while Transformers take more inference time and complexity to match ConvNet's performance while both ConvNext and Mixers are more efficient than their predecessors.

After considering these insights, our focus shifted towards leveraging the strengths of both ConvNext and Mixer. Combining a convolutional encoder for initial feature extraction with a mixer network in the bottleneck has demonstrated effectiveness. Convolutional layers excel in efficient low-level feature extraction, while mixers, following transformers, showcase strong representative abilities in extracting deep features, as also highlighted in Srinivas et al. (2021). We performed experiments where we slowly introduce Mixer in the bottleneck of ConvNext which can be seen in the following table 15 where the PSNR is reported on PortraitVideo dataset:

Table 2. Experiments on PortraitVideo dataset with difference configurations of ConvNext and Mixer at different encoder blocks.

ConvNext	Mixer	Latency (in ms)	PSNR
1,2,3,4,5	-	30.56	31.24
1,2,3	4,5	**15.58**	31.25
1,2	3,4,5	18.79	31.25
1	2,3,4,5	26.77	31.23
-	1,2,3,4,5	28.65	31.23
4,5	1,2,3	36.85	31.25

This illustrates our empirical journey towards the core design element of ReBotNet—a convolutional feature extractor followed by a mixer bottleneck. Notably, the last row presents an alternative configuration with early mixer layers and bottleneck ConvNext layers, resulting in sub-optimal performance. These analytical experiments were pivotal in finalizing the design choice of ReBotNet.

3. Configurations of ReBotNet

In the main paper, we mentioned we conducted experiments with different FLOPs regimes for all the methods. We did that by controlling the depth of the bottleneck and the embedding dimension of different methods to get the required FLOPs. In Tables 3, 4, and 5 we provide the exact configurations of ReBotNet - S,M, and L respectively.

[*]Parts of the work was done during an internship at Google.

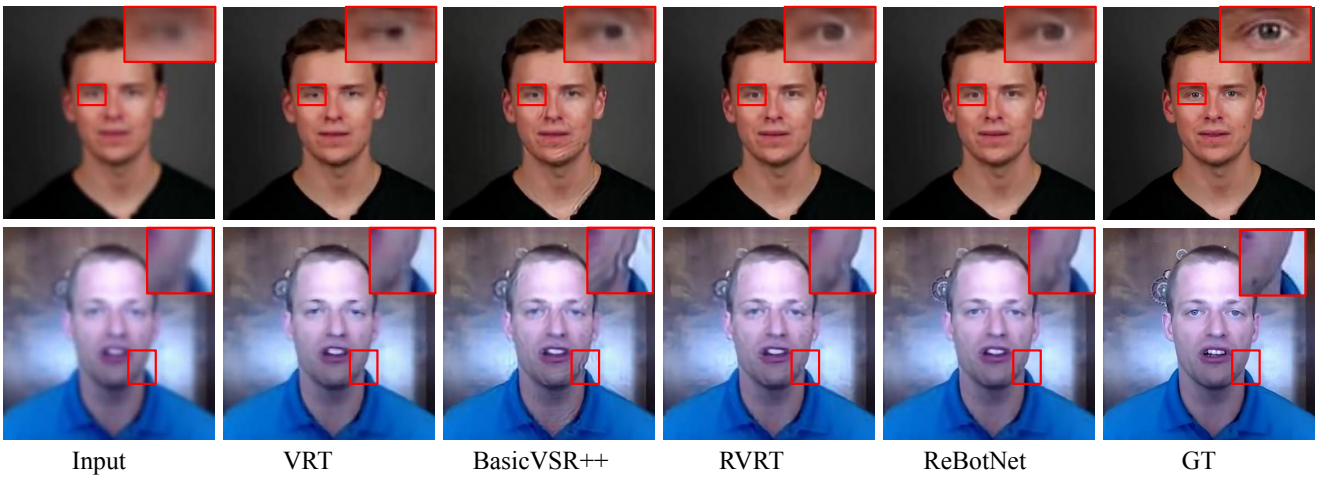

Figure 1. Qualitative Results on *PortraitVideo* dataset. Please zoom in for better visualization.

Figure 2. Qualitative Results on *FullVideo* dataset. Please zoom in for better visualization.

More analysis on the dependence of these parameters were provided in the main paper.

4. Configuration of Baselines

We used the publicly available codes for the original implementations of FastDVDNet, BasicVSR++, VRT, and RVRT; the results of which can be seen in Table 1 of the main paper. For the S,M and L configurations we use the same configurations of the original implementations but change the embedding dimensions. These changes have been illustrated in Tables 6, 7, and 8. OG means the original implementation. Note that RVRT does not have a S configuration as even with embedding dimensions of $[1, 1, 1]$, the FLOPs does not hit the range of 10 GFLOPs.

5. Experiments on Pure Mixers

We observed that MLP-Mixers tend to exhibit a noticeable decline in quality when applied directly for video en-

Table 3. Configuration of ReBotNet-S.

Block	Type	Value
Branch I	Number of Layers	4
	Depths per layer	[4,4,4,4]
	Embedding dimensions	[28,36,48,64]
Branch II	Patch size	1
	Embedding Dimension	256
Bottleneck	Depth	4
	Input Dimension	64
	Hidden Dimension	728

Table 4. Configuration of ReBotNet-M.

Block	Type	Value
Branch I	Number of Layers	4
	Depths per layer	[4,4,4,4]
	Embedding dimensions	[64,80,108,116]
Branch II	Patch size	1
	Embedding Dimension	256
Bottleneck	Depth	4
	Input Dimension	116
	Hidden Dimension	728

Table 5. Configuration of ReBotNet-L.

Block	Type	Value
Branch I	Number of Layers	4
	Depths per layer	[5,5,5,4]
	Embedding dimensions	[172,180,188,196]
Branch II	Patch size	1
	Embedding Dimension	256
Bottleneck	Depth	4
	Input Dimension	64
	Hidden Dimension	728

Table 6. Configurations of VRT.

Method	Embedding Dimension
VRT - S	[24,24,24,24,24,24,24,24,24,24]
VRT - M	[48,48,48,48,48,48,48,48,48,48]
VRT - L	[180,180,180,180,180,180,120,120,120,120]
VRT - OG	[180,180,180,180,180,180,120,120,120,120,120,120]

Table 7. Configurations of RVRT.

Method	Embedding Dimension
RVRT - S	-
RVRT - M	[36,36,36]
RVRT - L	[192,192,192]
RVRT - OG	[192,192,192]

hancement compared to transformer-based approaches. Using Mixers directly on large size images still takes a lot of compute and makes it difficult to achieve real-time speed.

Table 8. Configurations of FastDVDNet.

Method	Embedding Dimension
FastDVDNet - S	[32, 48, 72, 96]
FastDVDNet - M	[64, 80, 108, 116]
FastDVDNet - L	[96, 112, 132, 144]
FastDVDNet - OG	[80, 96, 132, 144]

In Table 9, we conduct an experiment where we take VRT as the base network and convert all the transformer blocks in it to MLP-Mixers. The experiment is conducted on the DVD dataset. It can be observed that the although the computation reduces, the performance also drops significantly. And still the computation is far away from obtaining a real-time FPS. This motivates us to work towards our design of ReBotNet as seen in the main paper.

Table 9. Experiment on pure mixers.

Method	PSNR	SSIM	GFLOPs	FPS
VRT	34.24	0.9651	2054.32	1
VRT (Mixers)	32.14	0.9429	1495.06	2

6. Degradations

In Table 10, we provide the detailed configurations of degradations that we use in PortraitVideo and FullVideo dataset. In all the rows where there is a range, we choose a random value in the range. To get the final degradation of a sample image at hand, we choose a random combination of the degradations from Table 10. These values were decided to emulate degradations possible in real-world and after consulting experts working in the field of video conferencing.

Table 10. Degradations used in PortraitVideo and FullVideo datasets.

Type of Degradation	Value
Eye Enlarge ratio	1.4
Blur kernel size	15
Kernel Isotropic Probability	0.5
Blur Sigma	[0.1,3]
Downsampling range	[0.8,2.5]
Noise amplitude	[0,0.1]
Compression Quality	[70,100]
Brightness	[0.8,1.1]
Contrast	[0.8,1.1]
Saturation	[0.8,1.1]
Hue	[-0.05,0.05]

7. Temporal Consistency

We would like to note that quantitatively evaluating temporal consistency is an actively researched field. With that being said, we report the difference in SSIM between consecutive frames to quantitatively evaluate the temporal consistency. The following table 17 is the mean SSIM difference across consecutive frames for all videos in the Portrait Video dataset. Differences in SSIM can give us details about how smooth the transitions are temporally where low difference means temporally consistent cases. It can be seen that the SSIM difference is less for the configuration including temporal branch proving the usefulness of our method.

Table 11. SSIM difference reported to understand the temporal consistency aspect of the introduced temporal branch.

Config	SSIM Difference
Ground Truth	0.104
Without Temp. Branch	0.158
With Temp. Branch	0.124

8. Limitations

i) While we do obtain real-time performance, there is still room of improvement in terms of performance to reach more pleasing results as visually one can still see a difference when we compare with high resolution ground truth. ii) There is potential to further increase the training data size to make sure the model works directly off-the-shelf. This would be essential to make the model work on everyday things like mobile phone videos. iii) We also have limitations with regard to the degradations we have used in the datasets. There is potential to test with more extreme degradations and check if it improves the real world performance on corner cases.