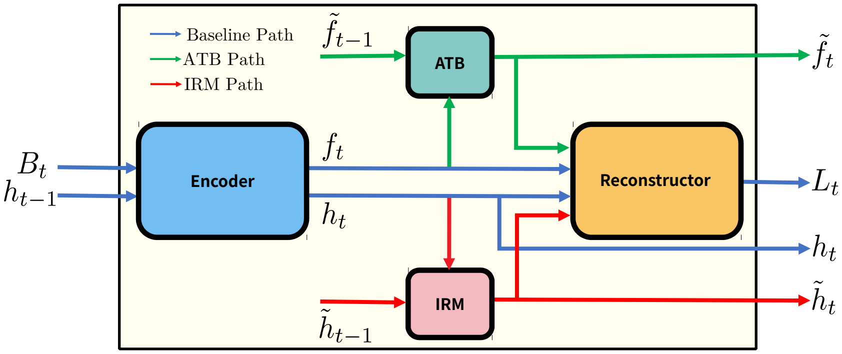
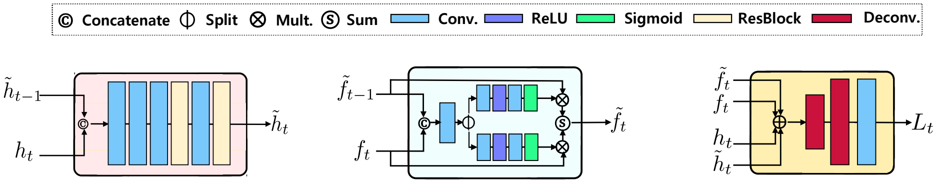
Supplementary Material *for* Recurrence-in-Recurrence Networks for Video Deblurring

Joonkyu Park
jkpark0825@snu.ac.kr
Seungjun Nah
seungjun.nah@gmail.com
Kyoung Mu Lee
kyoungmu@snu.ac.kr

ASRI, Department of ECE
Seoul National University
Korea



(a) Recurrence-in-Recurrence Network

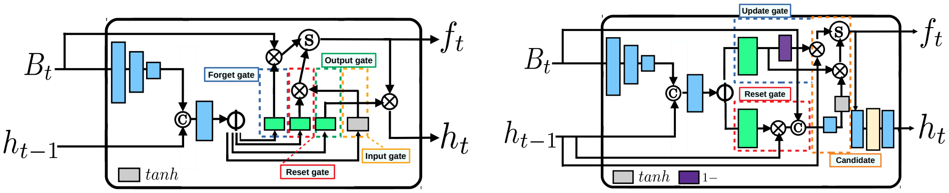


(b) Inner-Recurrence Module (c) Adaptive Temporal Blending (d) Reconstruction Module

Figure S1: The architecture of Recurrence-in-Recurrence Network and the components

S1 Introduction

In the main manuscript, we proposed recurrence-in-recurrence networks (RIRN) that serve as add-on modules for RNN-based video deblurring methods. In this supplementary material, we provide the experimental details and extensive quantitative and qualitative comparisons



(a) LSTM (b) GRU

Figure S2: The architecture of GRU and LSTM models.

of the deblurred results. In Section S2, the models used in the experiments in the main manuscript are described with the specifics. In Section S3, the design choice of IRM and ATB architecture for RIRN is justified by swapping each module with the other architecture. In Section S4, the effect of IRM and ATB are elaborated by applying each modules and both of them to the baseline architectures. In Section S5, the deblurred results are visually compared in the attached videos (Sec S5.1) and the captured frames (Sec S5.2).

S2 Model specification

In Figure S1a, the overview of our RIRN architecture is shown. The Inner-Recurrence Module (IRM) and Adaptive Temporal Blending (ATB) improves the baseline RNNs in terms of restoration accuracy. In the experimental comparisons, we used the state-of-the-art video deblurring methods, IFI-RNN [6], STRCNN [3], RDBN [7]. Also, we conducted experiments with LSTM [2] and GRU [1]-based models that are meant to control the RNN cell memory with gates to show our IRM and ATB could supplement the memory updating scheme.

S2.1 Encoder Architecture

For IFI-RNN [6], we used the simplest model, C1H1 with a single cell without the iterative operations. The number of feature channels equal to the original model. In STRCNN [3], we removed the DTB module to compare the effect with our ATB. We built the RDBN architecture by adapting the ESTRNN [7]. We removed the GSA module that fuses the saved features from multiple time steps to compare the effect in memory control with our RIRN. We used the $B_{15}C_{80}$ model using 15 RDB blocks with 80 feature channels.

In constructing the LSTM and GRU models for video deblurring, we modified IFI-RNN (C1H1). As IFI-RNN is a type of RNN without a gating function, we add the corresponding gates on IFI-RNN to construct LSTM and GRU. We employ LSTM and GRU model by adapting IFI-RNN architecture. The constructed architectures are shown in Figure S2. The detailed layer specifics are described in Table S1 for LSTM and Table S2 for GRU.

S2.2 IRM architecture

IRM architecture is shown in Figure S1b. The two input states are concatenated and then processed by convolutional layers and ResBlocks. The detailed specifics are shown in Table S3.

S2.3 ATB architecture

ATB generates the attention map for the input features and outputs the weighted sum of the features from the attention. The concatenated inputs are processed by a convolutional layer and then are split in half by the channel dimension. The following layers for each split branch generates the attention map that is to be multiplied with the corresponding input. The attended features are added to output the blended feature. The detailed specifics are shown in Table S4.

Table S1: LSTM architecture details. h, w are the number of height and width.

Module	layer	kernel	stride	output shape
B_t	input	-	-	$3 \times h \times w$
h_{t-1}	input	-	-	$20 \times h/4 \times w/4$
	conv	5×5	1	$20 \times h \times w$
	conv	5×5	2	$40 \times h/2 \times w/2$
	conv	5×5	2	$60 \times h/4 \times w/4$
	concat	-	-	$80 \times h/4 \times w/4$
	conv	5×5	1	$80 \times h/4 \times w/4$
	split	-	-	$(20 \times h/4 \times w/4) \times 4$
Forget gate	sigmoid	-	-	$20 \times h/4 \times w/4$
Input gate	tahn	-	-	$20 \times h/4 \times w/4$
Output gate	sigmoid	-	-	$20 \times h/4 \times w/4$
Reset gate	sigmoid	-	-	$20 \times h/4 \times w/4$
f_t	output	-	-	$60 \times h/4 \times w/4$
h_t	output	-	-	$20 \times h/4 \times w/4$

Table S2: GRU architecture details. h, w are the number of height and width.

Module	layer	kernel	stride	output shape
B_t	input	-	-	$3 \times h \times w$
h_{t-1}	input	-	-	$20 \times h/4 \times w/4$
	conv	5×5	1	$20 \times h \times w$
	conv	5×5	2	$40 \times h/2 \times w/2$
	conv	5×5	2	$60 \times h/4 \times w/4$
	concat	-	-	$80 \times h/4 \times w/4$
	conv	5×5	1	$80 \times h/4 \times w/4$
	split	-	-	$60 \times h/4 \times w/4, 20 \times h/4 \times w/4$
Update gate	sigmoid	-	-	$(60 \times h/4 \times w/4)$
Reset gate	sigmoid	-	-	$20 \times h/4 \times w/4$
	concat	-	-	$80 \times h/4 \times w/4$
Candidate	conv	5×5	1	$60 \times h/4 \times w/4$
Candidate	tanh	-	-	$60 \times h/4 \times w/4$
	conv	3×3	1	$20 \times h/4 \times w/4$
	ResBlock	3×3	1	$20 \times h/4 \times w/4$
	conv	3×3	1	$20 \times h/4 \times w/4$
f_t	output	-	-	$60 \times h/4 \times w/4$
h_t	output	-	-	$20 \times h/4 \times w/4$

Table S3: IRM architecture details. c, h, w are the number of channels, height, and width. c varies by the baseline model architecture.

Module	layer	kernel	stride	output shape
h_t	input	-	-	$c \times h \times w$
$\tilde{h}_{t-1}$	input	-	-	$c \times h \times w$
	concat	-	-	$2c \times h \times w$
	conv	3×3	1	$2c \times h \times w$
	conv	3×3	1	$2c \times h \times w$
	conv	3×3	1	$2c \times h \times w$
	ResBlock	3×3	1	$2c \times h \times w$
	conv	3×3	1	$c \times h \times w$
	ResBlock	3×3	1	$c \times h \times w$
$\tilde{h}_t$	output	-	-	$c \times h \times w$

Table S4: ATB architecture details. c, h, w are the number of channels, height, and width. c varies by the baseline model architecture.

Module	layer	kernel	stride	output shape
f_t	input	-	-	$c \times h \times w$
$\tilde{f}_{t-1}$	input	-	-	$c \times h \times w$
	concat	-	-	$2c \times h \times w$
	conv	3×3	1	$2c \times h \times w$
	split	-	-	$c \times h \times w$, $c \times h \times w$
Upper path	conv	3×3	1	$c \times h \times w$
	ReLU	-	-	$c \times h \times w$
	conv	1×1	1	$c \times h \times w$
	sigmoid	-	-	$c \times h \times w$
Lower path	conv	3×3	1	$c \times h \times w$
	ReLU	-	-	$c \times h \times w$
	conv	1×1	1	$c \times h \times w$
	sigmoid	-	-	$c \times h \times w$
w_t	-	-	-	$c \times h \times w$
$\tilde{w}_{t-1}$	-	-	-	$c \times h \times w$
$\tilde{f}_t$	output	-	-	$c \times h \times w$

Table S5: Application of IRM and ATB on existing RNN architectures

Architecture	GOPRO		REDS	
	PSNR	SSIM	PSNR	SSIM
IFI-RNN (C1H1) [6]	28.30	0.8668	30.01	0.8762
IFI-RNN + ATB + ATB	28.65	0.8812	30.75	0.8822
IFI-RNN + IRM +IRM	28.76	0.8874	30.95	0.8900
IFI-RNN + IRM + ATB	**29.14**	**0.8894**	**31.08**	**0.8905**

Table S6: Application of IRM and ATB on existing RNN architectures

Architecture	GOPRO		REDS	
	PSNR	SSIM	PSNR	SSIM
IFI-RNN (C1H1) [6]	28.30	0.8668	30.01	0.8762
IFI-RNN + ATB	28.65	0.8779	30.61	0.8800
IFI-RNN + IRM	28.88	0.8805	30.75	0.8845
IFI-RNN + IRM + ATB	**29.14**	**0.8894**	**31.08**	**0.8905**
STRCNN [3]	28.72	0.8460	30.23	0.8708
STRCNN + ATB	28.75	0.8554	30.43	0.8752
STRCNN + IRM	28.82	0.8602	30.66	0.8829
STRCNN + IRM + ATB	**28.87**	**0.8781**	**30.76**	**0.8902**
RDBN [7]	29.82	0.9043	32.29	0.9222
RDBN + ATB	29.87	0.9048	32.55	0.9221
RDBN + IRM	29.96	0.9086	32.44	0.9308
RDBN + IRM + ATB	**30.17**	**0.9120**	**32.71**	**0.9322**
GRU	25.11	0.7890	26.69	0.7956
GRU + ATB	25.49	0.7961	26.77	0.8043
GRU + IRM	26.03	0.8179	28.52	0.8338
GRU + IRM + ATB	**26.36**	**0.8217**	**28.60**	**0.8428**
LSTM	25.22	0.7948	26.87	0.8046
LSTM + ATB	25.33	0.8011	27.69	0.8198
LSTM + IRM	26.97	0.8291	28.75	0.8335
LSTM + IRM + ATB	**27.24**	**0.8400**	**29.12**	**0.8584**

S3 Design Ablation of RIRN

In RIRN, we proposed IRM to handle the long-range dependency of hidden states and ATB to improve the temporal blending of image features. To validate the validity of the designs of each modules for the corresponding temporal dependency range, we conduct ablation study in Table S5 by replacing each module with the other module. Compared with the cases the same sub-architecture is used to handle both the relation among the hidden states and among the image features by either the ATB or the IRM, our proposed RIRN showed the best restoration quality.

S4 Quantitative Comparison

In Table S6, we show the detailed comparison of video deblurring results by showing the effect of ATB, IRM. Each module consistently shows the improvement over the baseline models. The best performance is achieved when both the modules are used.

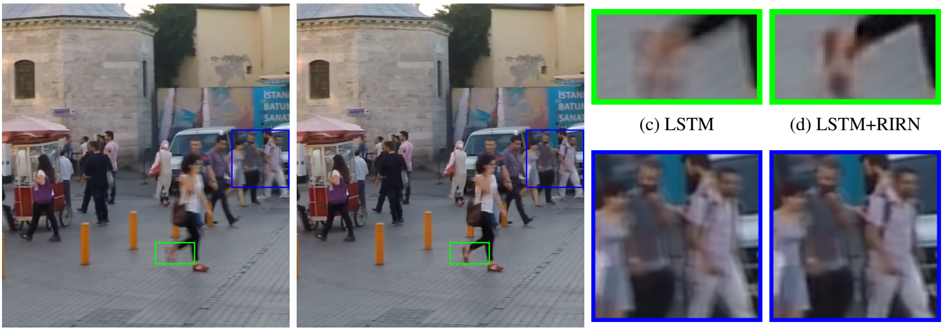
S5 Visual Comparison of Deblurred Videos

S5.1 Video Results

We present the videos showing the deblurred results with the blurry input in the supplementary videos. Please see the attached videos for the comprehensive comparison.
`GOPRO.mp4`:
Comparison between blur GOPRO video and deblurred video with RDBN+RIRN.
`REDS.mp4`:
Comparison between blur REDS video and deblurred video with RDBN+RIRN.
`REAL.mp4`:
Comparison between REAL blur video and deblurred video with STRCNN+RIRN.

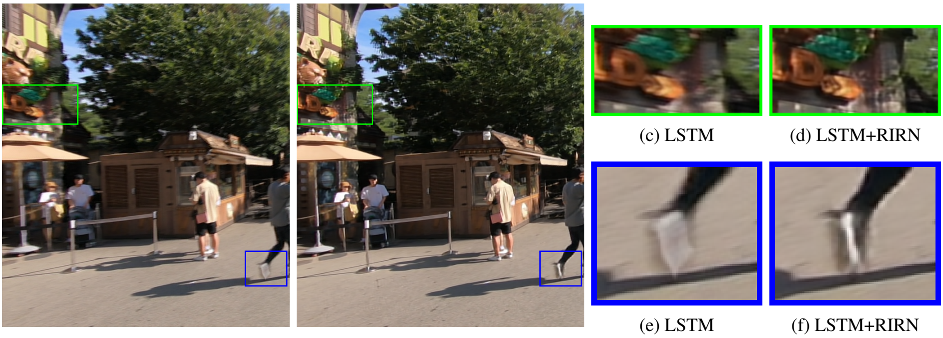
S5.2 Captured Results

More visual comparisons are elaborated in the below figures by showing the results on GOPRO [4], REDS [5] datasets and on the real videos we collected. In Figure S3, S4, and S5, LSTM and LSTM+RIRN are compared. In Figure S6, S7, and S8, GRU and GRU+RIRN are compared. In Figure S9, S10, and S11, STRCNN and STRCNN+RIRN are compared. In Figure S12, S13, and S14, IFI-RNN (C1H1) and IFI-RNN (C1H1)+RIRN are compared. In Figure S15, S16, and S17, RDBN and RDBN+RIRN are compared. RIRN consistently improves the visual quality of the deblurred videos.



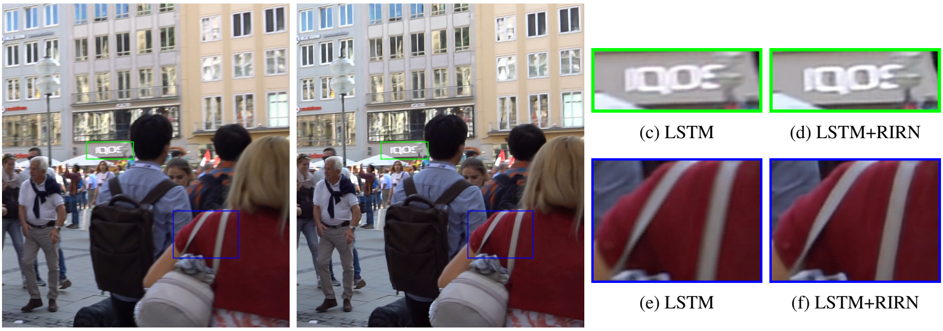
(a) LSTM (b) LSTM+RIRN (c) LSTM (d) LSTM+RIRN (e) LSTM (f) LSTM+RIRN

Figure S3: Visual comparison between LSTM and LSTM+RIRN on GOPRO dataset.



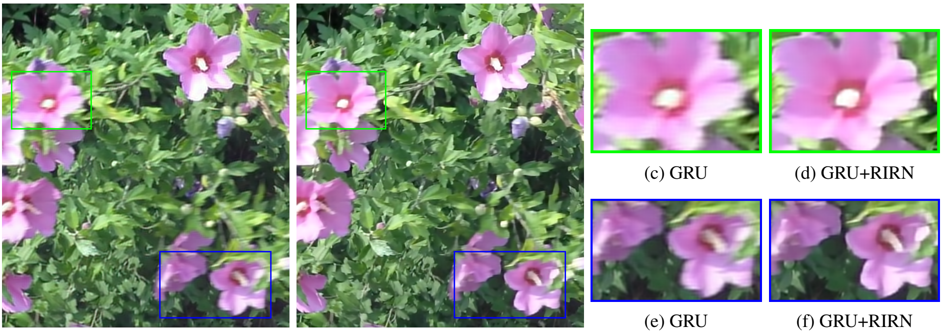
(a) LSTM (b) LSTM+RIRN (c) LSTM (d) LSTM+RIRN (e) LSTM (f) LSTM+RIRN

Figure S4: Visual comparison between LSTM and LSTM+RIRN on REDS dataset.



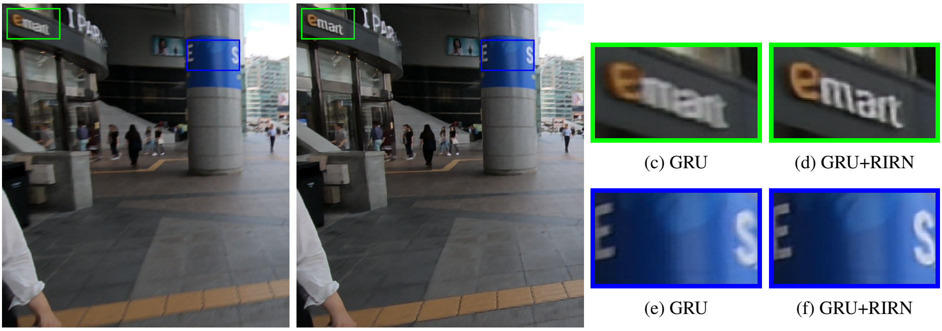
(a) LSTM (b) LSTM+RIRN (c) LSTM (d) LSTM+RIRN (e) LSTM (f) LSTM+RIRN

Figure S5: Visual comparison between LSTM and LSTM+RIRN on a real blurry video.



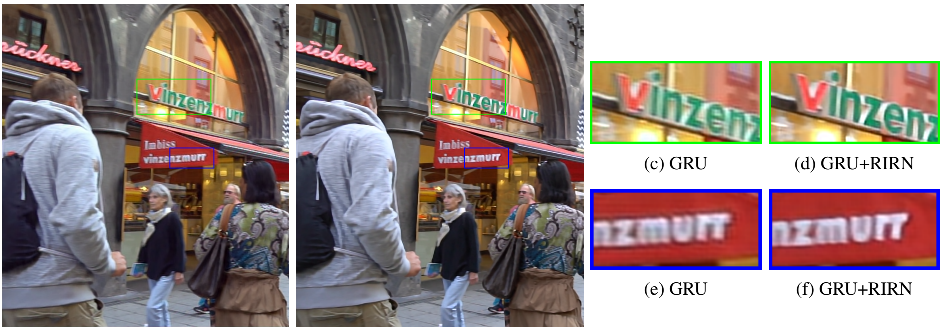
(a) GRU (b) GRU+RIRN (c) GRU (d) GRU+RIRN (e) GRU (f) GRU+RIRN

Figure S6: Visual comparison between GRU and GRU+RIRN on GOPRO dataset.



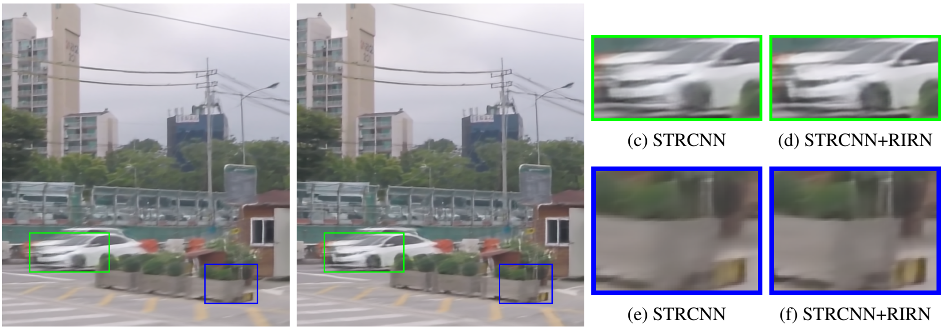
(a) GRU (b) GRU+RIRN (c) GRU (d) GRU+RIRN (e) GRU (f) GRU+RIRN

Figure S7: Visual comparison between GRU and GRU+RIRN on REDS dataset.



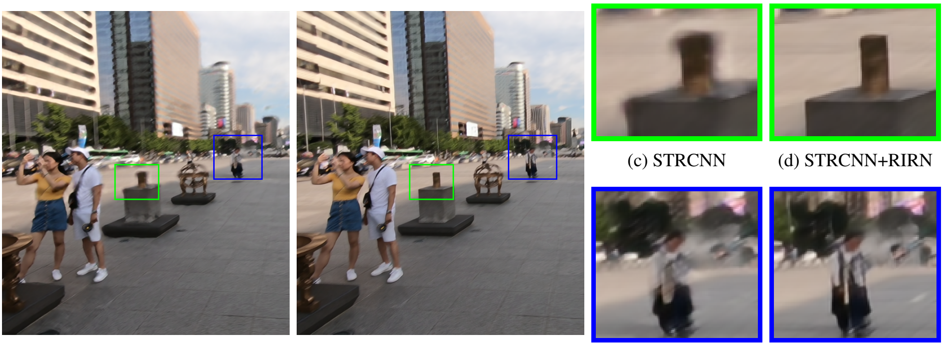
(a) GRU (b) GRU+RIRN (c) GRU (d) GRU+RIRN (e) GRU (f) GRU+RIRN

Figure S8: Visual comparison between GRU and GRU+RIRN on a real blurry video.



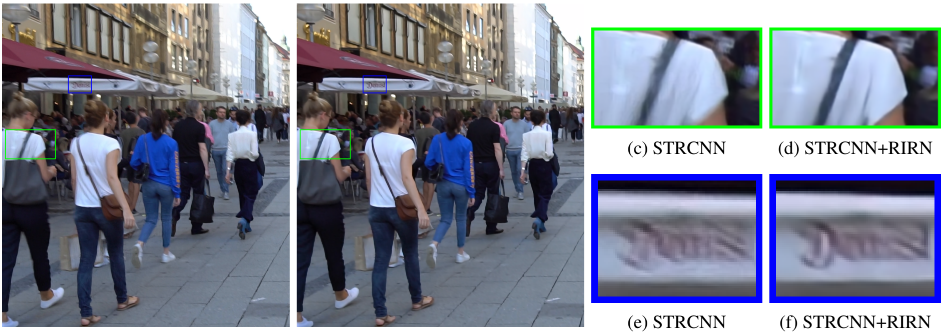
(a) STRCNN (b) STRCNN+RIRN (c) STRCNN (d) STRCNN+RIRN (e) STRCNN (f) STRCNN+RIRN

Figure S9: Visual comparison between STRCNN and STRCNN+RIRN on GOPRO dataset.



(a) STRCNN (b) STRCNN+RIRN (c) STRCNN (d) STRCNN+RIRN (e) STRCNN (f) STRCNN+RIRN

Figure S10: Visual comparison between STRCNN and STRCNN+RIRN on REDS dataset.



(a) STRCNN (b) STRCNN+RIRN (c) STRCNN (d) STRCNN+RIRN (e) STRCNN (f) STRCNN+RIRN

Figure S11: Visual comparison between STRCNN and STRCNN+RIRN on a real blurry video.

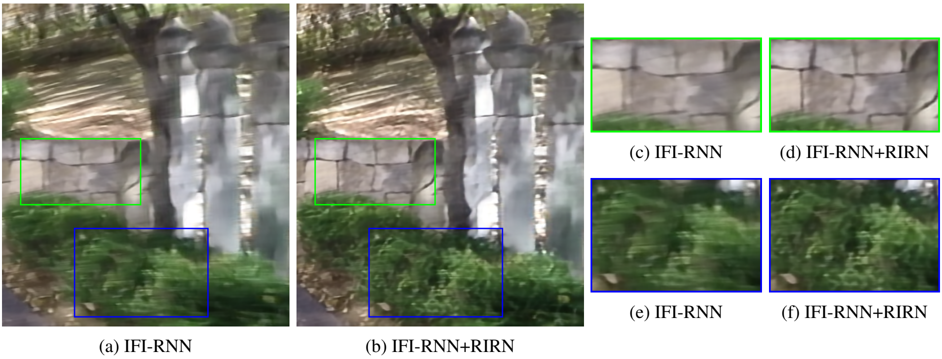

(a) IFI-RNN (b) IFI-RNN+RIRN (c) IFI-RNN (d) IFI-RNN+RIRN (e) IFI-RNN (f) IFI-RNN+RIRN

Figure S12: Visual comparison between IFI-RNN and IFI-RNN+RIRN on GOPRO dataset.

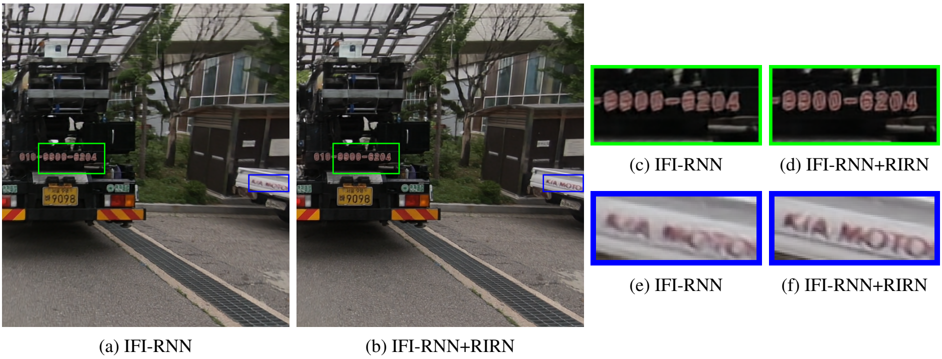

(a) IFI-RNN (b) IFI-RNN+RIRN (c) IFI-RNN (d) IFI-RNN+RIRN (e) IFI-RNN (f) IFI-RNN+RIRN

Figure S13: Visual comparison between IFI-RNN and IFI-RNN+RIRN on REDS dataset.

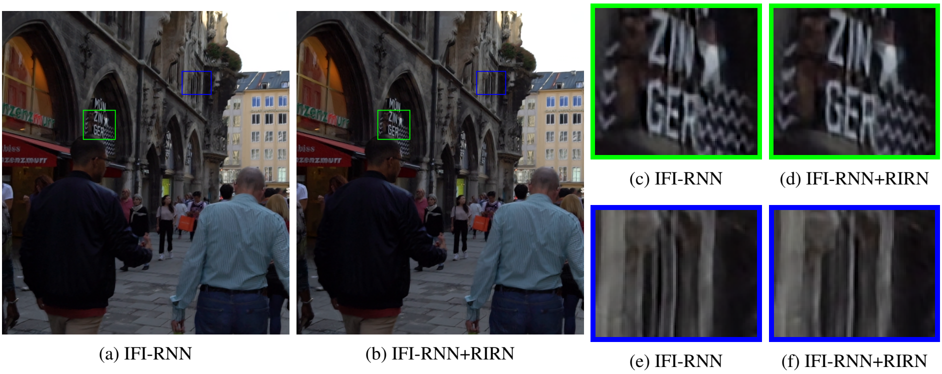
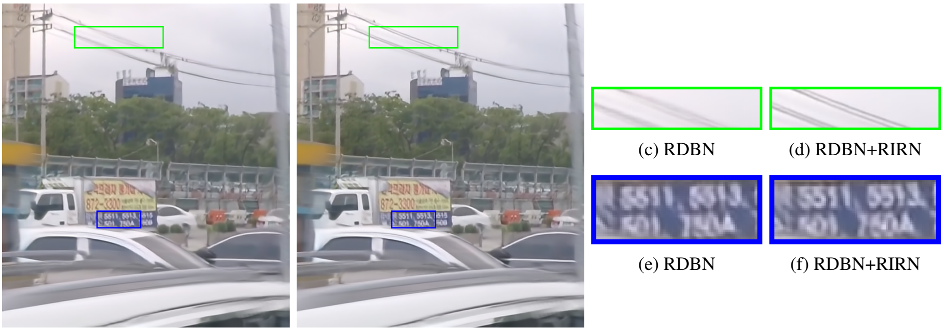
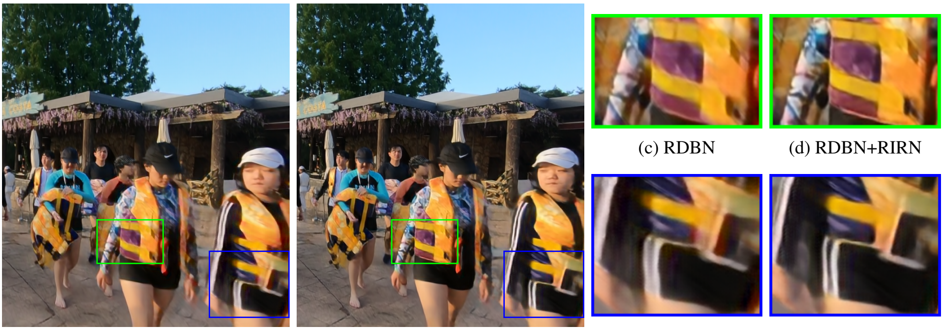

(a) IFI-RNN (b) IFI-RNN+RIRN (c) IFI-RNN (d) IFI-RNN+RIRN (e) IFI-RNN (f) IFI-RNN+RIRN

Figure S14: Visual comparison between IFI-RNN and IFI-RNN+RIRN on a real blurry video.



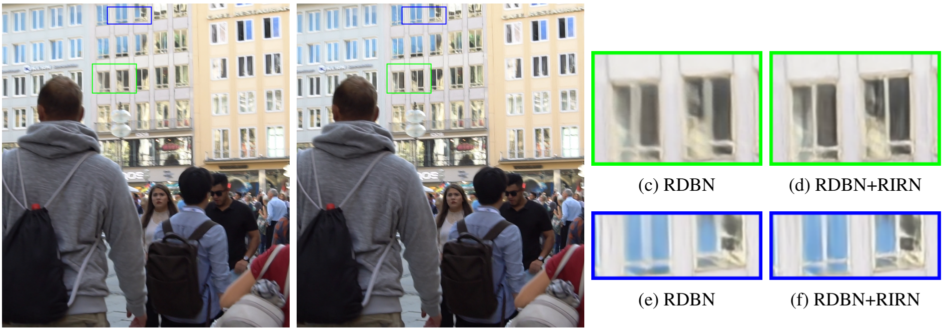
(a) RDBN (b) RDBN+RIRN (c) RDBN (d) RDBN+RIRN (e) RDBN (f) RDBN+RIRN

Figure S15: Visual comparison between RDBN and RDBN+RIRN on GOPRO dataset.



(a) RDBN (b) RDBN+RIRN (c) RDBN (d) RDBN+RIRN (e) RDBN (f) RDBN+RIRN

Figure S16: Visual comparison between RDBN and RDBN+RIRN on REDS dataset.



(a) RDBN (b) RDBN+RIRN (c) RDBN (d) RDBN+RIRN (e) RDBN (f) RDBN+RIRN

Figure S17: Visual comparison between RDBN and RDBN+RIRN on a real blurry video.

References

[1] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

[2] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[3] Tae Hyun Kim, Kyoung Mu Lee, Bernhard Schölkopf, and Michael Hirsch. Online video deblurring via dynamic temporal blending network. In *ICCV*, 2017.

[4] Seungjun Nah, Tae Hyun Kim, and Kyoung Mu Lee. Deep multi-scale convolutional neural network for dynamic scene deblurring. In *CVPR*, 2017.

[5] Seungjun Nah, Sungyong Baik, Seokil Hong, Gyeongsik Moon, Sanghyun Son, Radu Timofte, and Kyoung Mu Lee. NTIRE 2019 challenges on video deblurring and super-resolution: Dataset and study. In *CVPR Workshops*, 2019.

[6] Seungjun Nah, Sanghyun Son, and Kyoung Mu Lee. Recurrent neural networks with intra-frame iterations for video deblurring. In *CVPR*, 2019.

[7] Zhihang Zhong, Ye Gao, Yinqiang Zheng, and Bo Zheng. Efficient spatio-temporal recurrent neural network for video deblurring. In *ECCV*, 2020.