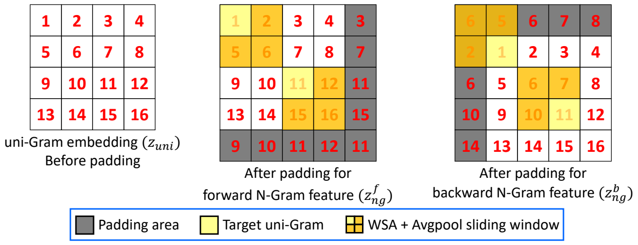
Supplementary Matrials for:
N-Gram in Swin Transformers for Efficient Lightweight Image Super-Resolution

Haram Choi[1] Jeongmin Lee[2] Jihoon Yang[1]*
[1]Department of Computer Science & Engineering, Sogang University [2]LG Innotek

A. Details of Other Components in NGswin

A.1. Sequentially Reflected Window Padding

As illustrated in the middle of Fig. A, $(N-1)$ size of paddings are applied at the lower-right side of uni-Gram embedding z_{uni} by sequentially reflected window padding (*seq-refl-win-pad*). Based on the outermost low/right windows, we use the upper/left $(N-1)$ rows/columns of windows as padding values. Consequently, *Sliding-WSA* produces the forward N-Gram feature z^f_{ng}. In turn, we can get the backward N-Gram feature z^b_{ng} by simply applying the same size of paddings on the upper-left side, as in the right of the figure. This allows some uni-Grams to interact with their padded neighbors, instead of trivial "zero" padding values. Our *seq-refl-win-pad* does not require extra Mult-Adds operations, because both zero padding and our padding method additionally give the same number of 32-bit float data to the input feature maps. We emphasize the advantage of *seq-refl-win-pad* in Sec. C.



1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16

1	2	3	4	3
5	6	7	8	7
9	10	11	12	11
13	14	15	16	15
9	10	11	12	11

6	5	6	7	8
2	1	2	3	4
6	5	6	7	8
10	9	10	11	12
14	13	14	15	16

uni-Gram embedding (z_{uni})
Before padding
After padding for
forward N-Gram feature (z^f_{ng})
After padding for
backward N-Gram feature (z^b_{ng})
▨ Padding area
🟨 Target uni-Gram
⊞ WSA + Avgpool sliding window

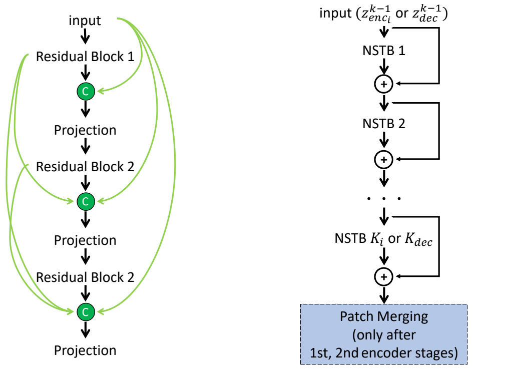
Figure A. Sequentially reflected window padding. N-Gram size N is 2. **(Left)** The uni-Gram embedding before padding. **(Middle)** Padding for the forward N-Gram feature z^f_{ng}. **(Right)** Padding for the backward N-Gram feature z^b_{ng}. As stated in Sec. 3.4, *sliding-WSA* weights for bi-directional N-Gram features are shared.

A.2. Within-Stage Residual Connections

While our across-stage pooling cascading follows the global cascading of CARN [2], the elements within a stage are residually connected [12]. Each NSTB and a *patch-merging* layer (except the third encoder stage and the decoder stage) in the encoder and decoder stages are residually connected. As described in Fig. B, this differs from the local cascading of CARN that employed dense connections [14]. Note that we did not specifically state the input to the decoder in the main content. z^{k-1}_{dec} in the figure is the input to k-th $(1{\leq}k{\leq}\mathcal{K}_{dec})$ NSTB in the decoder stage. The corresponding mapping function $\mathcal{F}^k_{dec}$ is formulated as:

$$z^k_{dec} = \mathcal{F}^k_{dec}(z^{k-1}_{dec}),\ z^k_{dec} \in \mathbb{R}^{HW \times D},$$

where z^0_{dec} equals $z_{scdp} + z^{\mathcal{K}_1}_{enc_1}$ from SCDP bottleneck and the last NSTB in the first encoder stage (Sec. 3.5). Also, $z_{dec} = \mathrm{LayerNorm}(z^{\mathcal{K}_{dec}}_{dec}) \in \mathbb{R}^{HW \times D}$.



(a) Local cascading (b) within-stage residual connection

Figure B. Comparison of local cascading and within-stage residual connections. **(a)** In local cascading of CARN, the resiudal blocks are densely connected. **(b)** In our within-stage residual connection, NSTBs and *patch-merging* are residually connected.

A.3. Reconstruction Module

The only difference between the ×2, ×3, and ×4 models is the reconstruction module. As depicted in Fig. C, we vary the output channels of the first convolution and the scale factor of the pixel-shuffler. The last convolutional layer is the difference from other methods [2, 15, 20, 21], as previously mentioned in Sec. 3.3. The input to this module is $z_s + z_{dec}$ with global skip-connection as stated in the main content (z_s results from the shallow module).

*Corresponding author.

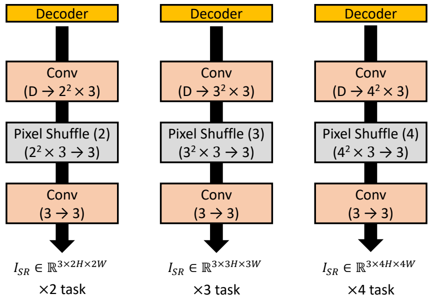


Figure C. Comparison of the reconstruction modules for different SR tasks. A parenthesis ($a \rightarrow b$) indicates change of channels (network dimension) from a to b. The other parenthesis (r) in Pixel Shuffle block indicates a scale factor (*e.g.*, $\times 4$).

Table A. Summary of learning strategies we find performed better with NGswin. Our findings are not absolute truths but just suggestions for the future works.

Method	Better	Worse
×3, ×4 training	warm-start [22]	scratch
std in normalization	from data [11]	1.0 [20, 21]
de-normalization position	before loss [21]	after loss
lr decay	half [21]	cosine [26]
weight decay [27]	no	yes
gradient clipping [28, 29]	no	yes
layer-wise *lr* decay [7]	no	yes
dropout [13]	no	yes
drop-path [19]	no	yes

B. Experimental Setup Details and Findings

In this section, we explain experimental settings and our findings from the results of various learning strategies. Since there are not an abundance of studies primarily focusing on training strategy itself for SR, we hope future researchers are able to gain insight from our findings. We summarize our findings in Tab. A. Although our findings in this section are not absolute truths, they can be helpful considerations for future research.

Model Architecture. The number of NSTBs in the encoder and decoder, $\{\mathcal{K}_1, \mathcal{K}_2, \mathcal{K}_3, \mathcal{K}_{dec}\}$, is set to $\{6, 4, 4, 6\}$. The number of WSA heads (for *sliding-WSA* and Swin Transformer's WSA) in each stage equals $\{\mathcal{K}_1, \mathcal{K}_2, \mathcal{K}_3, \mathcal{K}_{dec}\}$. We set the network dimension (channel) D, hidden dimension of FFN (feed-forward network) after Swin Transformer's WSA, window size M, and N-Gram size N to 64, 128, 8, and 2, respectively. The shift size is 4, *i.e.* $\lfloor \frac{M}{2} \rfloor$ same as in Swin V1 and V2 [24, 25]. The activation functions in FFN and after depth-wise convolution of SCDP bottleneck are GELU non-linearity. Also, we use LeakyReLU non-linearity after the iterative max-poolings of the pixel-shuffle [31] step in the bottleneck. For the other components not mentioned, there are no activation functions.

Training Details. We implemented the model configurations, training pipeline, and evaluation procedure by PyTorch [30] on 4 NVIDIA TITAN Xp GPUs. The batch size and training epochs were 64 and 500. We used Adam [16] optimizer with $\{\beta_1, \beta_2, \epsilon\} = \{0.9, 0.999, 10^{-8}\}$ for training from scratch (×2 task) and warm-start before whole fine-tuning (×3, ×4 tasks). For whole fine-tuning phase, AdamW [27] was utilized with the same hyper-parameters above. The learning rate (*lr*) was initialized as 0.0004 and decayed by half (half-decay) after $\{200, 300, 400, 425, 450, 475\}$ epochs. At the start of the training, we placed 20 warmup epochs [10] that linearly increased *lr* from 0.0 to initialized *lr* (10^{-4}).

Warm-Start. We trained NGswin and SwinIR-NG from scratch (×2) and by warm-start (×3, ×4) [22], as mentioned in Sec. 4.1. The warm-start scheme lasts for 300 epochs. This strategy, therefore, needed short training times. Warm-start was processed as follows: Loading the pre-trained weight on ×2, we froze all layers except the reconstruction module, and trained this module for 50 epochs (warm-start epoch). In this phase, *lr* was kept as a constant (*i.e.*, 0.0004). After that, the whole parameters of the network were fine-tuned by back-propagation (whole fine-tuning) for 250 epochs. We placed 10 warmup epochs at the start of whole fine-tuning. In whole fine-tuning, *lr* was halved after $\{50, 100, 150, 175, 200, 225\}$ epochs. We compared SwinIR-NG trained by warm-start scheme to the scratch one in Tab. Bc to show the merits of this strategy.

Dataset. We never used any extra datasets other than 800 images from DIV2K [1]. Each data point in the training dataset was repeated 80 times in an epoch to maximize the merits of random-cropping (64×64), following ELAN [36]. The random horizontal flip and rotation of 90°, 180°, 270° augmented the training data. We converted all images including the test data to *".npy"* (numpy) files with the *uint8* data type, for faster loading and efficient memory usage.

Normalization. We normalized the training data using the means and standard-deviations (std) of 800 LR images on RGB channels matching each task. Expressly, it was the same as standardization. The outputs of the reconstruction module were de-normalized (inverse of normalization), and used for calculating L_1 pixel-loss. Although we trained the models with different normalization strategies such as 1.0 std and not de-normalizing before computing loss, those strategies fell behind the default strategy. On the other hand, SwinIR-NG was trained with 1.0 std, following SwinIR paper [20].

Learning Rate Decay. We observed that cosine learning rate decay (cosine-decay) [26] did not perform well on SR tasks. It was because the underfitting (not overfitting) is a crucial issue to SR [22]. Interestingly, it differs from the high-level vision tasks such as classification, object detection, and semantic segmentation. We hypothesize that the cosine-decay reduces the *lr* faster than the half-decay (keep-

Table B. Other ablation studies.

(a) Ablation study on Swin Transformer version.

Swin ver.	Scale	Mult-Adds	#Params	Set5	Set14	BSD100	Urban100	Manga109
V1	×2	140.41G	998,176	37.99 / 0.9606	33.71 / 0.9192	32.20 / 0.9000	32.28 / 0.9301	38.69 / 0.9770
V2			998,384	**38.05 / 0.9610**	**33.79 / 0.9199**	**32.27 / 0.9008**	**32.53 / 0.9324**	**38.97 / 0.9777**
V1	×3	66.56G	1,006,831	34.42 / 0.9273	30.44 / 0.8445	29.13 / 0.8066	28.35 / 0.8569	33.66 / 0.9456
V2			1,007,039	**34.52 / 0.9282**	**30.53 / 0.8456**	**29.19 / 0.8078**	**28.52 / 0.8603**	**33.89 / 0.9470**
V1	×4	36.44G	1,018,948	32.20 / 0.8946	28.69 / 0.7836	27.61 / 0.7380	26.26 / 0.7916	30.53 / 0.9090
V2			1,019,156	**32.33 / 0.8963**	**28.78 / 0.7859**	**27.66 / 0.7396**	**26.45 / 0.7963**	**30.80 / 0.9128**

(b) Ablation study on padding method. The comparative model is NGswin.

Method	Scale	Set5	Set14	BSD100	Urban100	Manga109
zero-pad	×2	37.82 / 0.9599	33.38 / 0.9160	32.06 / 0.8983	31.58 / 0.9231	38.10 / 0.9759
seq-refl-win-pad		**38.05 / 0.9610**	**33.79 / 0.9199**	**32.27 / 0.9008**	**32.53 / 0.9324**	**38.97 / 0.9777**
zero-pad	×3	34.14 / 0.9249	30.22 / 0.8400	28.99 / 0.8030	27.74 / 0.8439	33.01 / 0.9410
seq-refl-win-pad		**34.52 / 0.9282**	**30.53 / 0.8456**	**29.19 / 0.8078**	**28.52 / 0.8603**	**33.89 / 0.9470**
zero-pad	×4	31.90 / 0.8906	28.45 / 0.7782	27.47 / 0.7332	25.72 / 0.7740	29.89 / 0.9016
seq-refl-win-pad		**32.33 / 0.8963**	**28.78 / 0.7859**	**27.66 / 0.7396**	**26.45 / 0.7963**	**30.80 / 0.9128**

(c) Ablation study on warm-start. The comparative model is SwinIR-NG.

Method	Scale	Set5	Set14	BSD100	Urban100	Manga109
scratch	×3	**34.65** / 0.9291	**30.59 / 0.8471**	29.23 / **0.8090**	28.71 / 0.8636	34.17 / 0.9485
warm-start		34.64 / **0.9293**	30.58 / **0.8471**	**29.24 / 0.8090**	**28.75 / 0.8639**	**34.22 / 0.9488**
scratch	×4	**32.45** / 0.8979	28.80 / 0.7867	27.71 / 0.7413	26.51 / 0.7992	31.02 / 0.9158
warm-start		32.44 / **0.8980**	**28.83 / 0.7870**	**27.73 / 0.7418**	**26.61 / 0.8010**	**31.09 / 0.9161**

ing a constant for long phases) and leads to the underfitting. However, we also observed that a decay point that was too early or too late ended up with the wrong converging point and decreased performances.

Regularization. We found that the SR tasks were hampered by the regularization strategies, such as weight decay [27], gradient clipping [28, 29], and layer-wise *lr* decay [7]. We compared a non-regularization strategy with the methods with 0.05 weight decay or 5.0 gradient clipping. However, these strategies dropped the performance. Similarly, while layer-wise *lr* decay improved the performance of high-level vision tasks when fine-tuning Transformer models [3, 11], our models could not learn the representations well with that regularization. This is also because SR tasks suffer from underfitting unlike recognition tasks.

Dropout. Considering that the crucial issue of SR tasks is underfitting, the dropout also had a negative effect on our work. Although we utilized the different dropout [13] and drop-path [19] rates, they were not good for NGswin. However, a recent work [17] has demonstrated the appropriate dropout strategy could improve SR performance.

C. Other Ablation Studies

Swin Transformer Version. Tab. Ba demonstrates the superiority of SwinV2 over SwinV1 for SR tasks of NGswin. As mentioned in [24], it is because dot-product self-attention of SwinV1 tends to make a few pixel pairs dominate the trained attention maps. But SR tasks need not some certain but neighbor pixels to recover degraded regions and reconstruct HR images. Since normalization is inherent in the cosine similarity of SwinV2, scaled-cosine self-attention can hinder some certain pixels from hugely affecting reconstruction tasks.

Padding. We investigated the impacts of *seq-refl-win-pad* in Tab. Bb. The trivial zero-padding (zero-pad) often conveys meaningless values to the feature maps, which causes several degraded regions to interact with empty pixels. It was a severely adverse method for uni-Gram embedding that had significantly low resolution (8×8, 4×4, 2×2). However, *seq-refl-win-pad* could give non-zero neighbors —some neighbors from other directions, to be precise— to the uni-Grams that were insufficient to neighbors. As a result, the networks could learn more meaningful representations, compared to zero-pad. Even the zero-pad approach was worse than the model without the N-Gram context. Mult-Adds operations and the number of parameters of the models were unchanged.

Warm-Start. It is obvious that the warm-start strategy requires much shorter training time than training from scratch (scratch). As explained in Sec. B, the scratch and warm-start scheme lasted for 500 and 300 epochs. Meanwhile, the 50 warm-start epochs that only updated the reconstruction module spent fractional times. Therefore, the warm-start scheme was about twice faster than scratch. However, we wondered if this scheme would also be better when it comes to SR performances as in RCAN-it [22]. Tab. Bc shows that SwinIR-NG trained by warm-start scheme recorded higher scores than scratch. Therefore, the warm-start training strategy is superior to scratch in terms of both time resource and performances for SR tasks.

Other Benchmarks. Due to the page limit in the main content, we did not report the results of ablation studies on some SR benchmark test datasets. In Tab. C, we posted those results including benchmarks already shown in Sec. 4.3. The results consistently showed the positive effect of each proposed (or employed) approach.

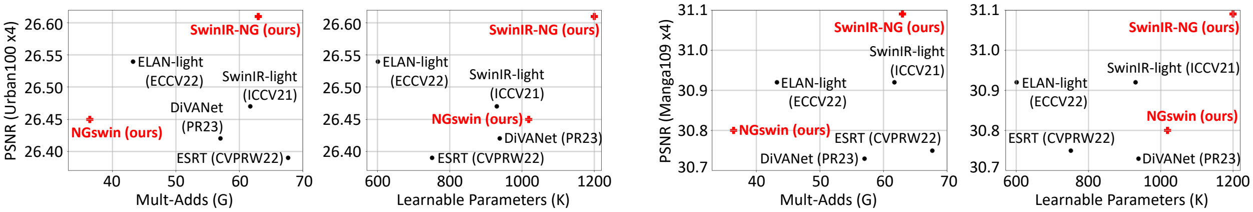


Figure D. Trade-off between performance and efficiency. The parameter sizes of NGswin and SwinIR-NG are 4.04MB and 4.74MB.

D. More Visual Comparisons

We supply more visual comparisons with other models in Figs. E, F, G and H. Also, more comparisons of the models with *vs*. without the N-Gram context are visualized in Figs. I and J.

E. Discussions and Limitations

In this section, we discuss the characteristics of our novel methods including the novelty and limitations. Moreover, we reflect on other tasks that are not addressed in this paper but can be developed by our N-Gram context. Finally, it is considered how this work can be extended to further improve our methods and cover broader tasks.

Methods. Fig. D illustrates the trade-off between performance and efficiency of our models and the best lightweight SR methods. NGswin has the fewest Mult-Adds operations and SwinIR-NG presents the best performance.

[NGswin] As shown in Sec. 3.5 and Tab. 6, our proposed SCDP bottleneck compensated the performance loss of the hierarchical encoder. However, one may doubt whether NGswin can be improved by abandoning the hierarchical architecture, as in ELAN-light and SwinIR-light. Of course, the non-hierarchical structure would significantly lead to performance gains. However, the following approximation shows the influence of training input resolution hw of a single NSTB on Mult-Adds of $\times r$ task:

$$\mathrm{MultAdds(NSTB)} \approx (10 \times hw/2^{12}/(\frac{r}{2})^2)\mathrm{G}.$$

Therefore, if there were no *patch-merging* layers, the eight NSTBs in the 2nd and 3rd encoder stages would have increased the operations by about 17G for the ×4 task.

[SwinIR-NG] Compared with SwinIR-light (*w/o* N-Gram), SwinIR-NG (*w/* N-Gram) needs a small number of extra operations to establish state-of-the-art lightweight SR. The parameters were also kept as a tolerable size (4.74MB) for semiconductor system, as stated in Sec. 1. However, it is a limitation that SwinIR-NG requires more parameters and operations than ELAN-light [36]. These results are due to our intention of focusing on improved performance.

[N-Gram Context] Our N-Gram context differs from recent attention mechanisms proposed for the efficient self-attention (SA). First, while our *sliding-WSA* produces the average correlations in the spatial space, channel attention (CA) [37] computed SA in the channel space, which was employed in Restormer [34] and NAFNet [6]. Second, group-wise multi-scale self-attention (GMSA) proposed by ELAN-light [36] divided the feature maps into K groups to avoid intensive operations. In contrast, the dimensionality reduction of our channel-reducing group convolution (uni-Gram embedding) decreases the time complexity of *sliding-WSA*. Third, cross-shaped window (CSWin) [9] enlarged the receptive field of SA by splitting multi-heads horizontally and vertically, then computed SA in each multi-head group. Recently, Cross Aggregation Transformer (CAT) [38] adopted the similar SA strategy for multiple image restoration tasks with a large model size. Whereas, we calculate SA within N^2 uni-Gram embeddings. The weight sharing for the bi-directional N-Gram features also broadens the receptive field. As a result, the receptive field of *sliding-WSA* is expanded $2N^2$ times. Lastly, the parameters of the N-Gram context can be further reduced by properly adopting and varying other methods.

[SwinV2] Most recently, Swin2SR [8] adapted SwinV2 like NGswin. Unlike our model, Swin2SR employed a continuous relative position bias [24]. It is compelling that Swin2SR demonstrated a potential that SwinIR-NG could be improved by SwinV2, as in our Tab. Ba. However, Swin2SR was trained with 3,450 images from a merged dataset of DIV2K and Flickr2K [32], which were even more than our 800 training images. Likewise, the Swin2SR paper only reported the results of the ×2 lightweight SR task. Therefore, despite its remarkable performance, we excluded Swin2SR from Tab. 3 for fair comparison.

Other Tasks. In this paper, we worked on the super-resolution of the bicubic LR images. Recently, some researchers studied the blind SR [23], where the input LR images are from unknown degradation. In addition, other low-level vision tasks, such as deblurring, denoising, deraining, and JPEG artifact reduction, were developed by attention mechanisms [5, 33–35]. Since these image restoration tasks also need the contextual information of distorted regions like the bicubic SR, our model introducing N-Gram to image would be helpful. Secondarily, we visualized how our work can boost high-level vision tasks, such as classification (CIFAR10 [18]) and ST-VQA (Scene Text Visual Question Answering) [4] in Figs. K, L and M.

Table C. The results of ablation studies on entire benchmarks. The tables in the parenthesis are the corresponding ones of Sec. 4.3. The results in **bold** are the best of each comparative content. PSNR / SSIM are reported.

(a) N-Gram context (Tab. 4).

NGswin without *vs*. with N-Gram

N-Gram	Scale	Mult-Adds	#Params	Set5	Set14	BSD100	Urban100	Manga109
w/o	×2	138.20G	750K	**38.05** / 0.9609	33.70 / 0.9194	32.25 / 0.9006	32.39 / 0.9304	38.86 / 0.9775
w/		**140.41G**	**998K**	**38.05 / 0.9610**	**33.79 / 0.9199**	**32.27 / 0.9008**	**32.53 / 0.9324**	**38.97 / 0.9777**
w/o	×3	65.53G	759K	**34.53** / 0.9281	30.48 / 0.8451	29.15 / 0.8073	28.37 / 0.8573	33.81 / 0.9464
w/		**66.56G**	**1,007K**	34.52 / **0.9282**	**30.53 / 0.8456**	**29.19 / 0.8078**	**28.52 / 0.8603**	**33.89 / 0.9470**
w/o	×4	35.89G	771K	32.34 / 0.8963	28.70 / 0.7844	27.63 / 0.7390	26.25 / 0.7918	30.70 / 0.9123
w/o (channel up)		53.71G	1,189K	32.37 / **0.8973**	28.75 / 0.7854	27.65 / 0.7396	26.28 / 0.7927	30.73 / 0.9129
w/o (depth up)		47.88G	1,061K	**32.40** / 0.8967	28.75 / 0.7853	**27.66 / 0.7398**	26.37 / 0.7946	30.78 / **0.9133**
w/		**36.44G**	**1,019K**	32.33 / 0.8963	**28.78 / 0.7859**	**27.66** / 0.7396	**26.45 / 0.7963**	**30.80** / 0.9128

HNCT *vs*. HNCT-NG

N-Gram	Scale	Mult-Adds	#Params	Set5	Set14	BSD100	Urban100	Manga109
w/o	×2	82.39G	357K	38.08 / 0.9608	**33.65** / 0.9182	32.22 / 0.9001	32.22 / 0.9294	38.87 / **0.9774**
w/		**83.19G**	**424K**	**38.10 / 0.9610**	33.64 / **0.9195**	**32.25 / 0.9006**	**32.35 / 0.9306**	**38.94 / 0.9774**
w/o	×3	37.78G	363K	34.47 / 0.9275	30.44 / 0.8439	29.15 / 0.8067	28.28 / 0.8557	**33.81** / 0.9459
w/		**38.14G**	**431K**	**34.48 / 0.9280**	**30.48 / 0.8450**	**29.16 / 0.8074**	**28.38 / 0.8573**	**33.81 / 0.9464**
w/o	×4	22.01G	373K	32.31 / 0.8957	28.71 / 0.7834	27.63 / 0.7381	26.20 / 0.7896	30.70 / 0.9112
w/		**22.21G**	**440K**	**32.32 / 0.8960**	**28.72 / 0.7846**	**27.65 / 0.7391**	**26.23 / 0.7912**	**30.71 / 0.9114**

(b) N-Gram directions and interaction (Tab. 5). The second best results are in underline.

Direction	Type	Mult-Adds	#Params	Set5	Set14	BSD100	Urban100	Manga109
1	WSA	152.41G	1,238,056	38.05 / **0.9610**	33.78 / 0.9198	32.26 / 0.9006	**32.54** / 0.9322	38.90 / **0.9777**
4	WSA	139.56G	935,272	**38.07** / 0.9609	33.76 / 0.9197	32.25 / 0.9007	32.52 / 0.9317	38.92 / 0.9776
1	CNN	139.80G	1,327,528	38.04 / **0.9610**	33.77 / 0.9197	32.25 / 0.9005	32.45 / 0.9316	38.86 / 0.9775
2	CNN	139.38G	998,568	38.04 / **0.9610**	**33.83 / 0.9203**	32.26 / 0.9007	**32.54** / 0.9321	38.90 / 0.9776
4	CNN	139.17G	936,488	38.02 / 0.9609	33.77 / 0.9178	32.26 / 0.9006	32.52 / 0.9320	38.93 / **0.9777**
2	**WSA**	**140.41G**	**998,384**	38.05 / **0.9610**	33.79 / 0.9199	**32.27 / 0.9008**	32.53 / **0.9324**	**38.97 / 0.9777**

(c) Extra stages and SCDP bottleneck (Tab. 6).

Stages	SCDP	Scale	Mult-Adds	#Params	Set5	Set14	BSD100	Urban100	Manga109
extra	*w/o*	×2	87.98G	997K	38.02 / 0.9607	33.71 / 0.9193	32.20 / 0.8999	32.28 / 0.9298	38.72 / 0.9773
default	*w/o*		138.88G	992K	**38.08** / 0.9609	**33.81 / 0.9199**	32.24 / 0.9005	32.48 / 0.9321	38.92 / 0.9776
default	***w/***		**140.41G**	**998K**	38.05 / **0.9610**	33.79 / **0.9199**	**32.27 / 0.9008**	**32.53 / 0.9324**	**38.97 / 0.9777**
extra	*w/o*	×3	42.10G	1,006K	34.38 / 0.9272	30.43 / 0.8437	29.11 / 0.8060	28.33 / 0.8562	33.67 / 0.9453
default	*w/o*		65.85G	1,001K	34.47 / 0.9277	30.49 / 0.8454	29.17 / 0.8073	28.47 / 0.8596	33.81 / 0.9464
default	***w/***		**66.56G**	**1,007K**	**34.52 / 0.9282**	**30.53 / 0.8456**	**29.19 / 0.8078**	**28.52 / 0.8603**	**33.89 / 0.9470**
extra	*w/o*	×4	23.33G	1,018K	32.17 / 0.8943	28.65 / 0.7827	27.59 / 0.7369	26.22 / 0.7900	30.46 / 0.9090
default	*w/o*		36.06G	1,013K	32.29 / 0.8957	28.73 / 0.7849	27.64 / 0.7391	26.38 / 0.7954	30.71 / 0.9121
default	***w/***		**36.44G**	**1,019K**	**32.33 / 0.8963**	**28.78 / 0.7859**	**27.66 / 0.7396**	**26.45 / 0.7963**	**30.80 / 0.9128**

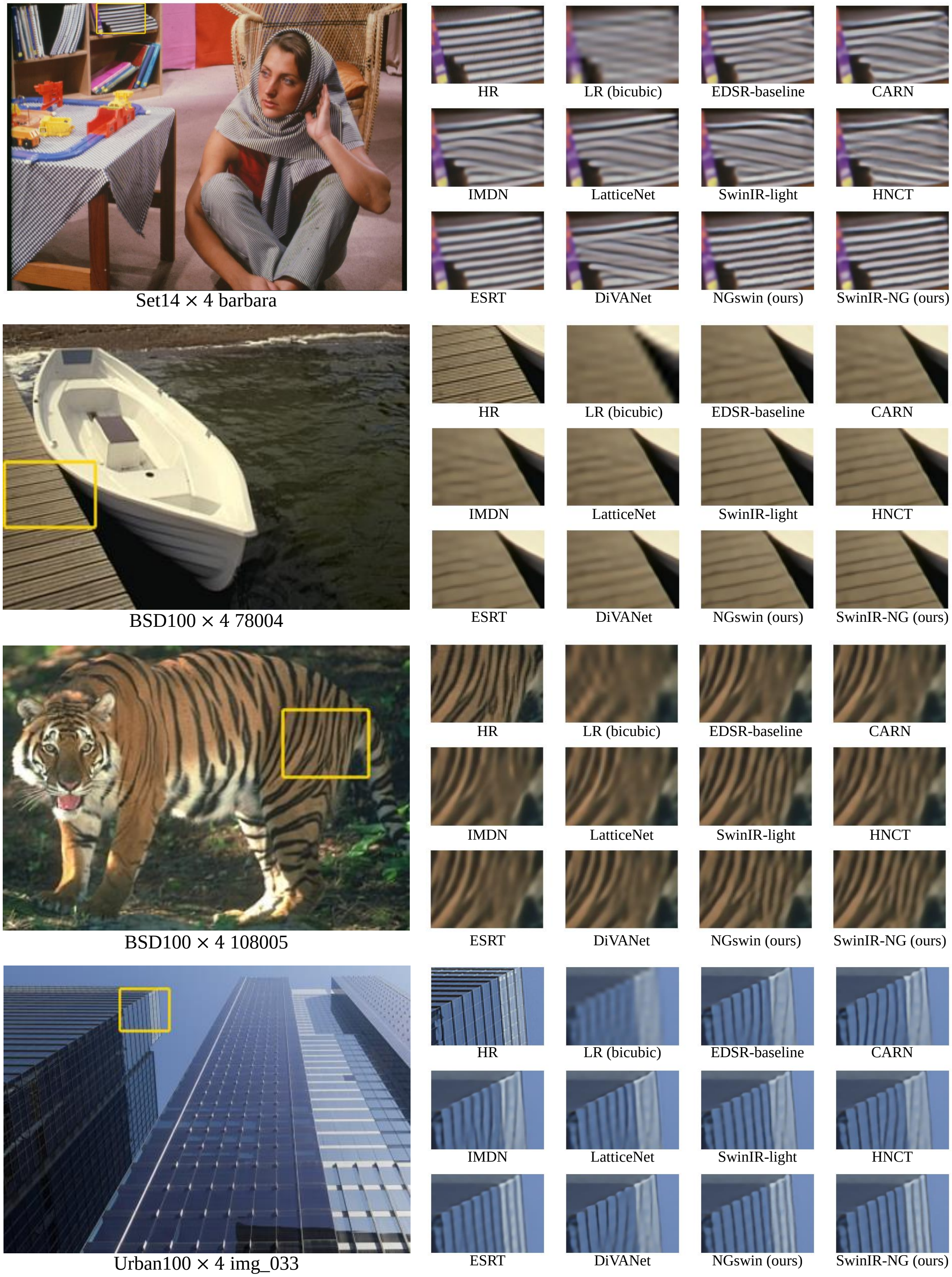


Figure E. Visual comparisons $(\times 4)$. “LR (bicubic)” indicates the low-resolution input images from bicubic interpolation.

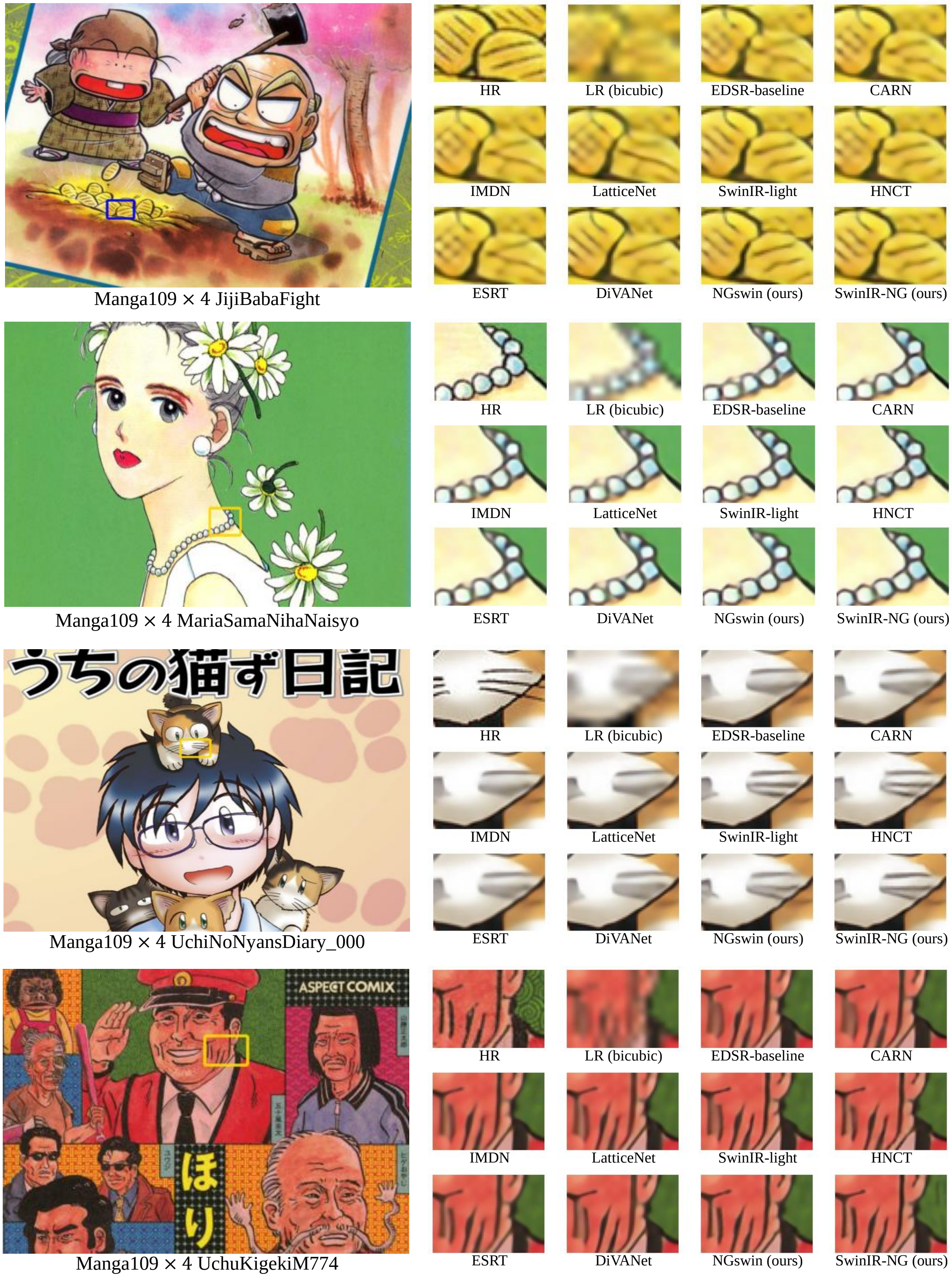
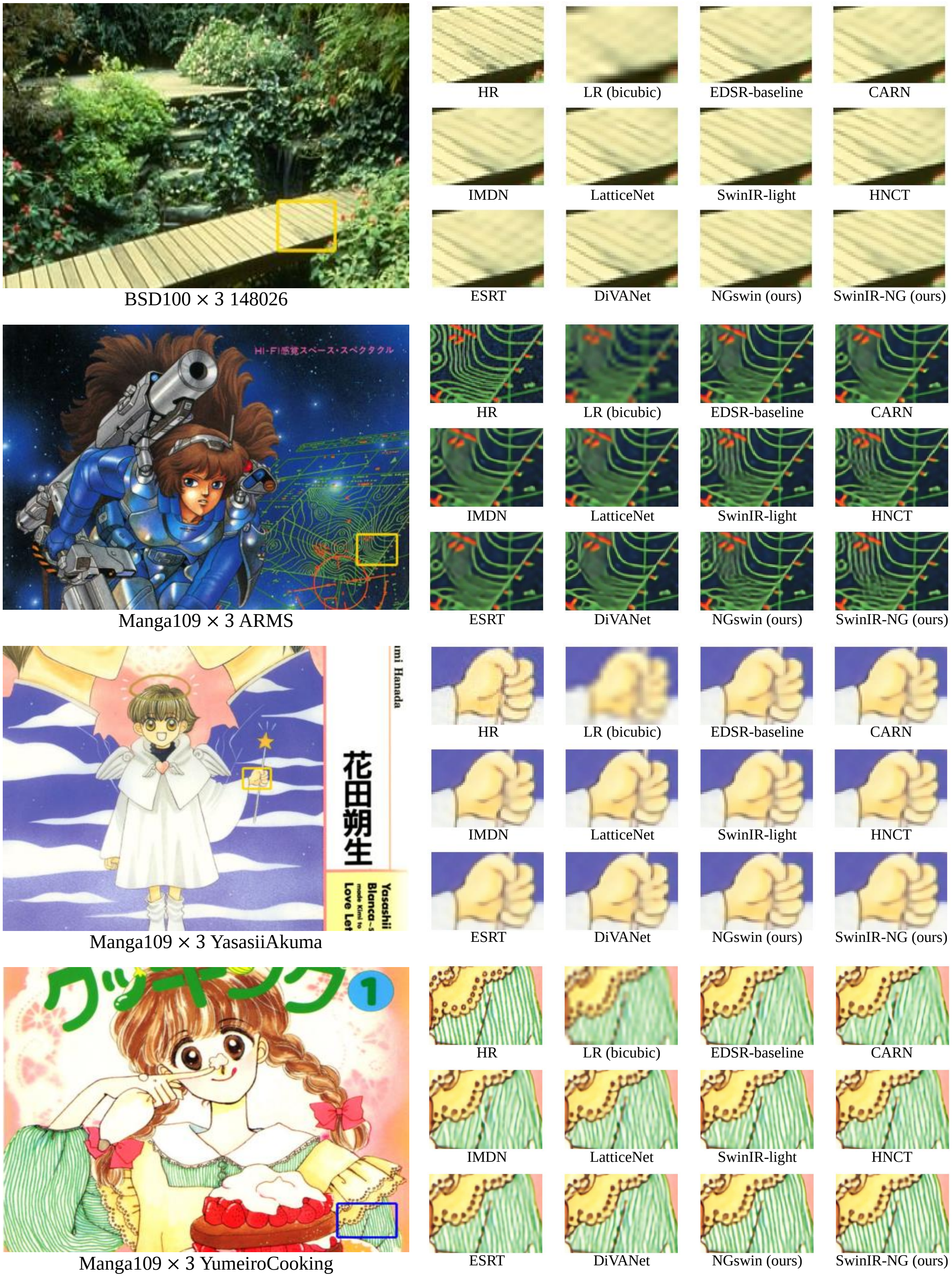


Figure F. Visual comparisons ($\times 4$). "LR (bicubic)" indicates the low-resolution input images from bicubic interpolation.

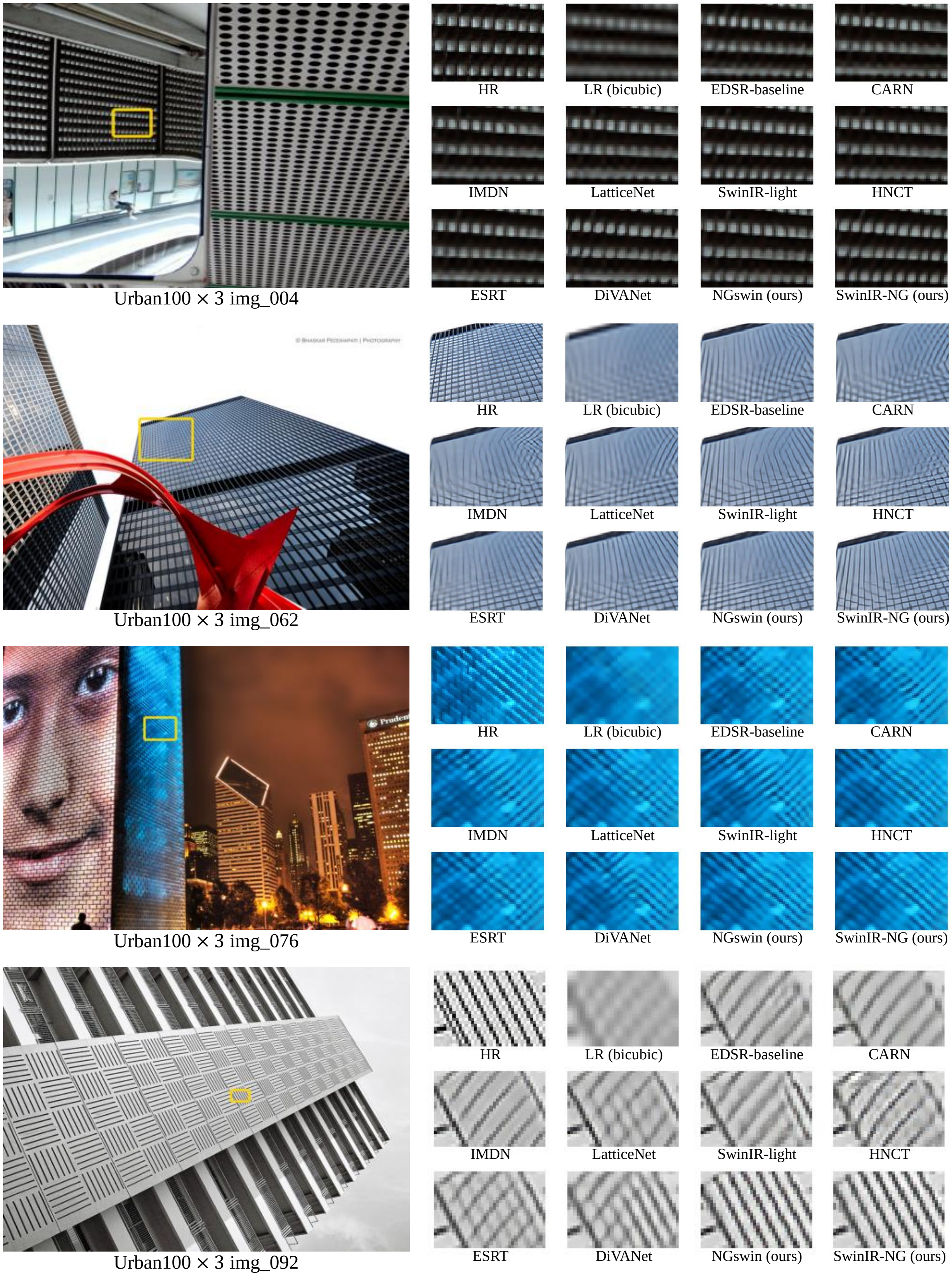

BSD100 × 3 148026

Manga109 × 3 ARMS

Manga109 × 3 YasasiiAkuma

Manga109 × 3 YumeiroCooking

Figure G. Visual comparisons (×3). “LR (bicubic)” indicates the low-resolution input images from bicubic interpolation.



Urban100 × 3 img_004

Urban100 × 3 img_062

Urban100 × 3 img_076

Urban100 × 3 img_092

Figure H. Visual comparisons ($\times 3$). “LR (bicubic)” indicates the low-resolution input images from bicubic interpolation.

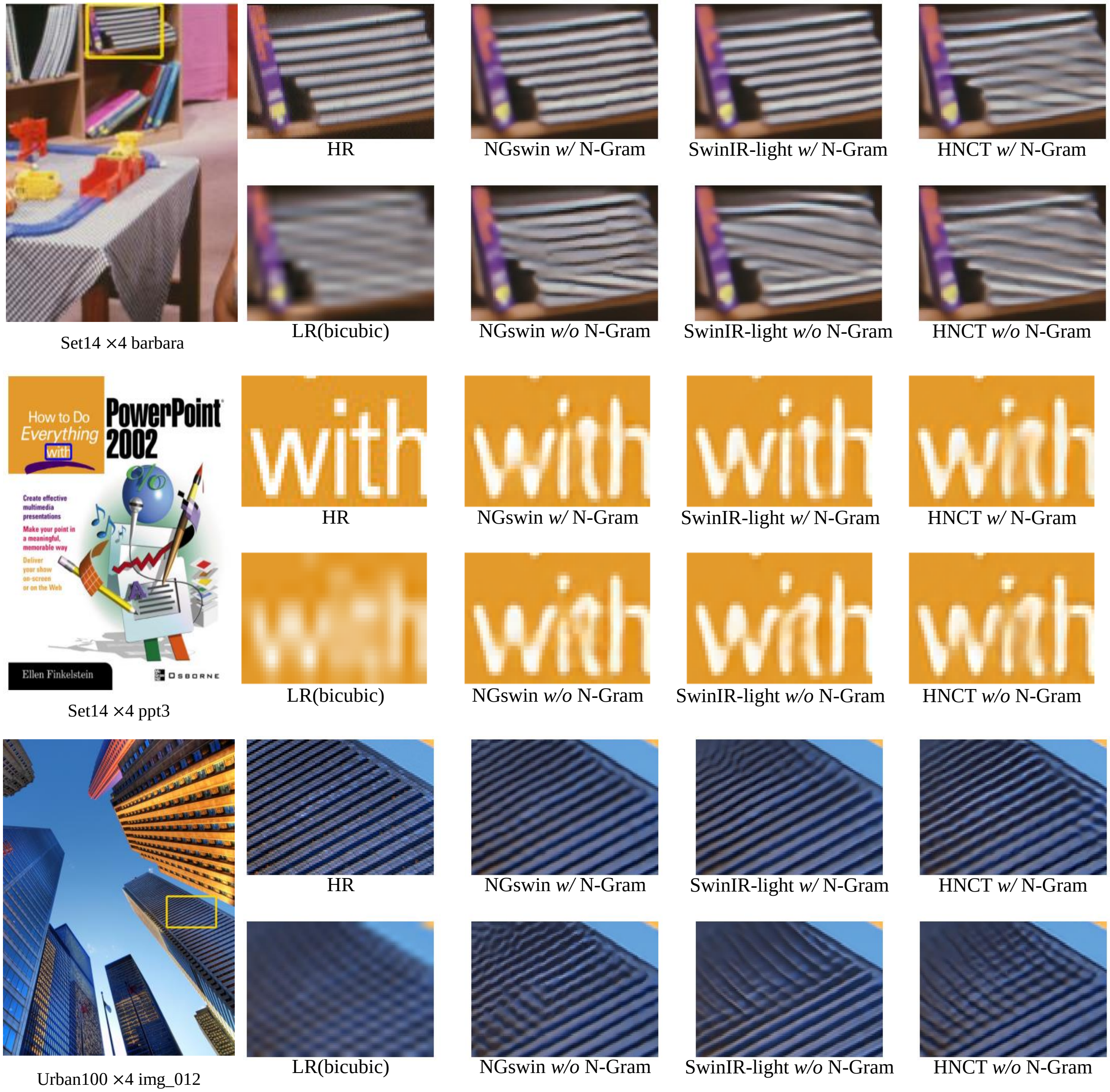


Figure I. Visual comparisons of the models with *vs*. without the N-Gram context (×4). “LR (bicubic)” indicates the low-resolution input images from bicubic interpolation.

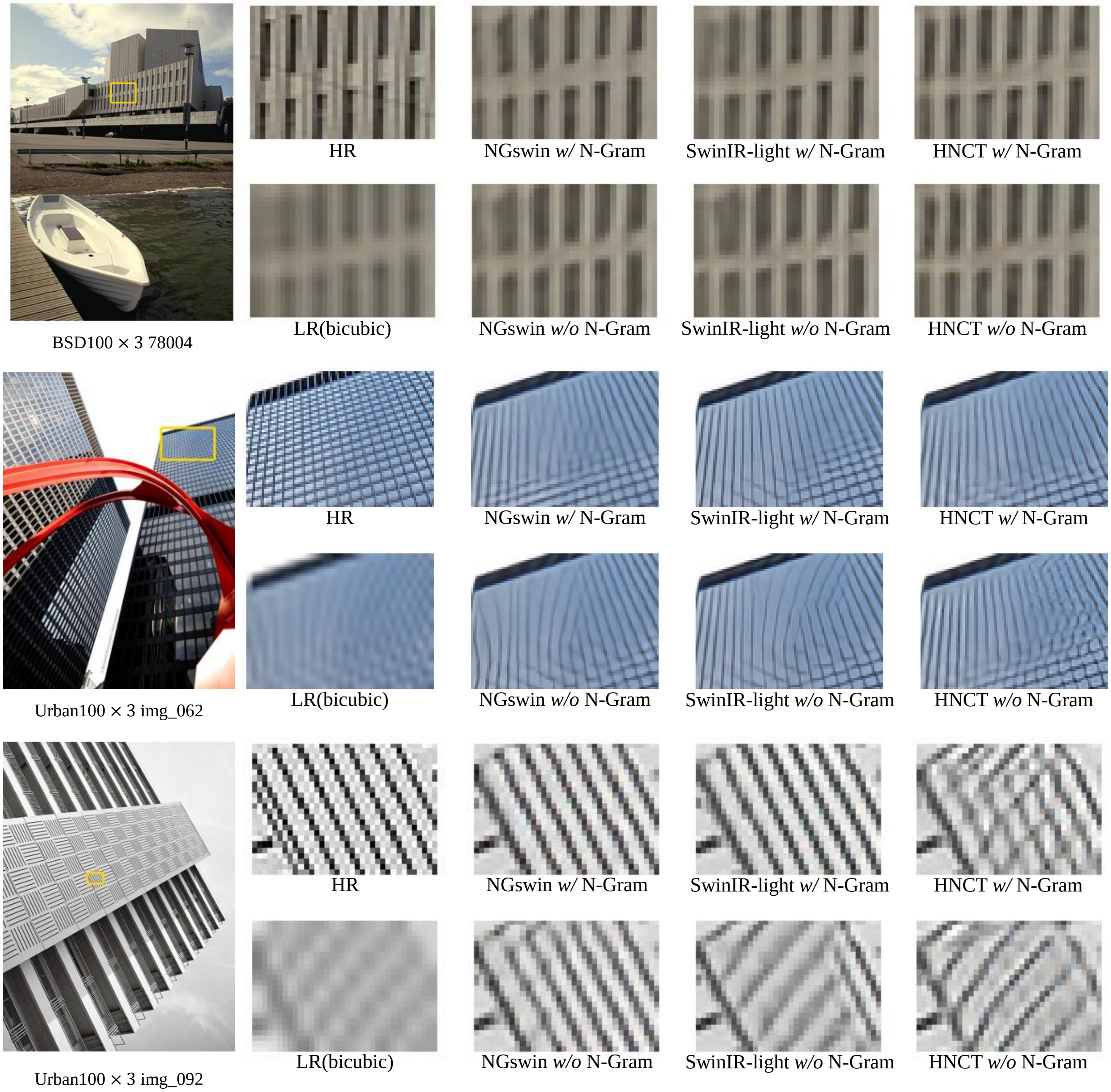


Figure J. Visual comparisons of the models with *vs*. without the N-Gram context (×3). "LR (bicubic)" indicates the low-resolution input images from bicubic interpolation.

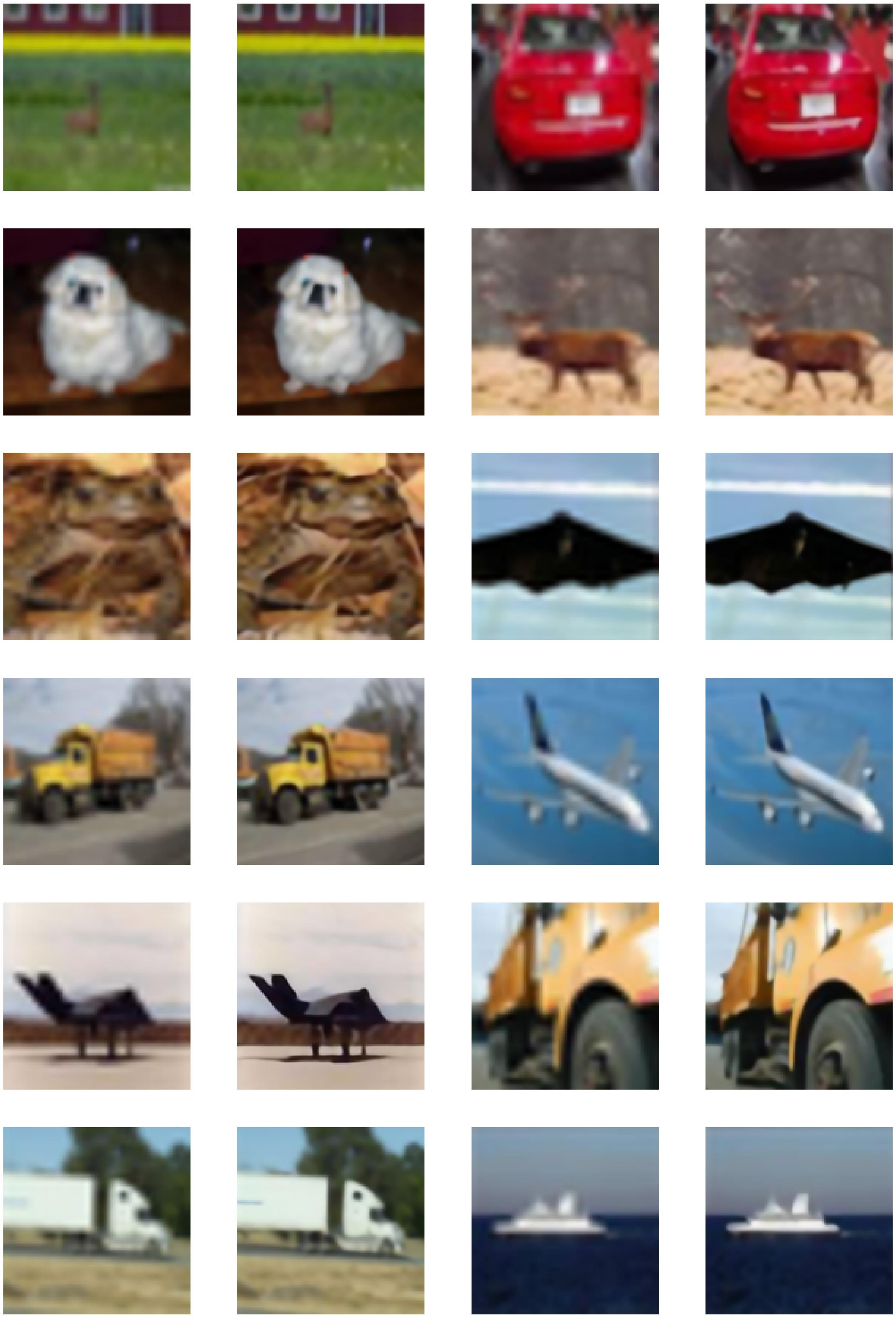


Figure K. Visual results with SwinIR-NG on CIFAR10 [18] (×4). Our technique may boost classification tasks with the sharper edges of super-resolution results. The 1st and 3rd columns are LR (bicubic) images. The 2nd and 4th columns are from SwinIR-NG. As this figure is secondary provision, we do not compare ours with other models.

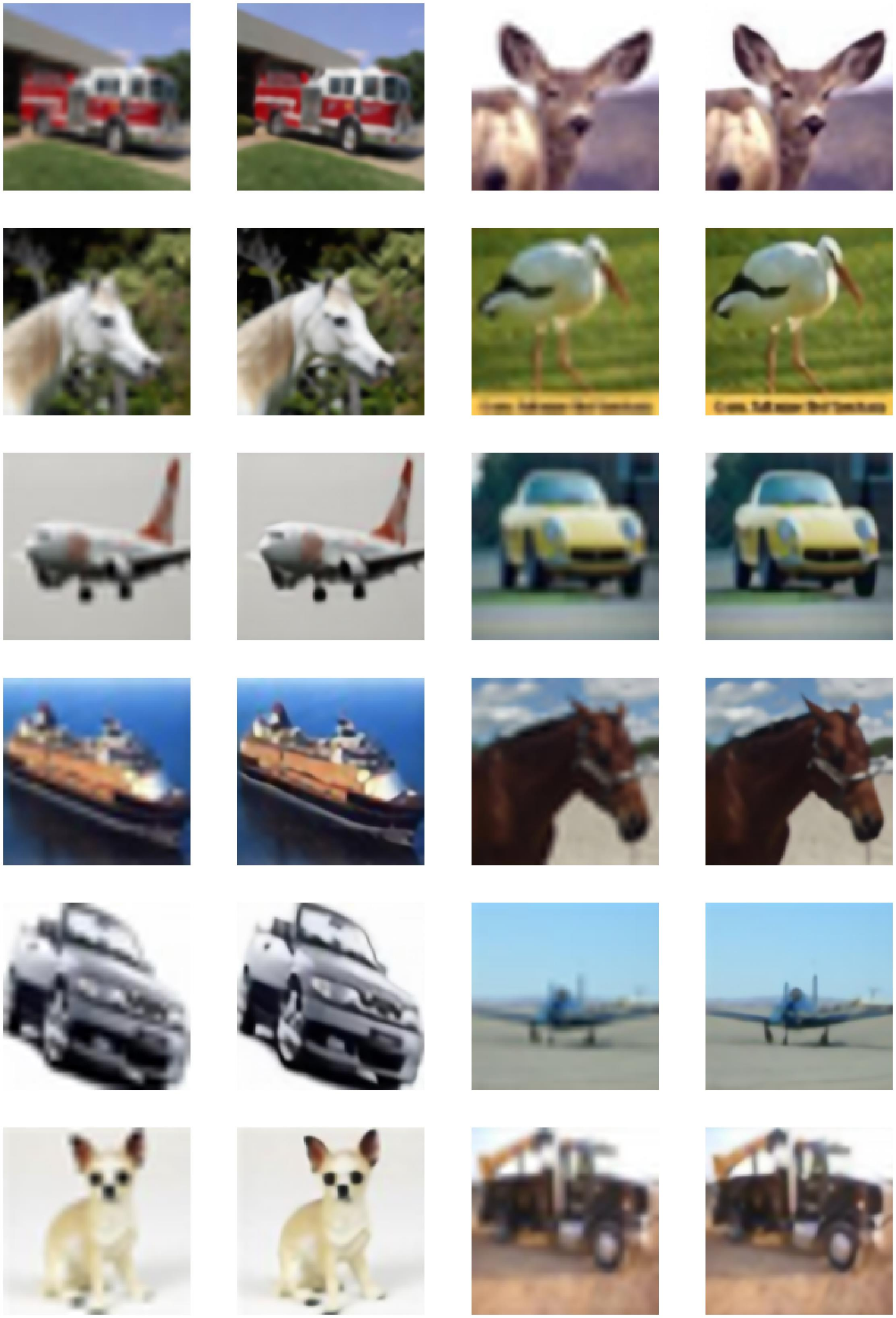
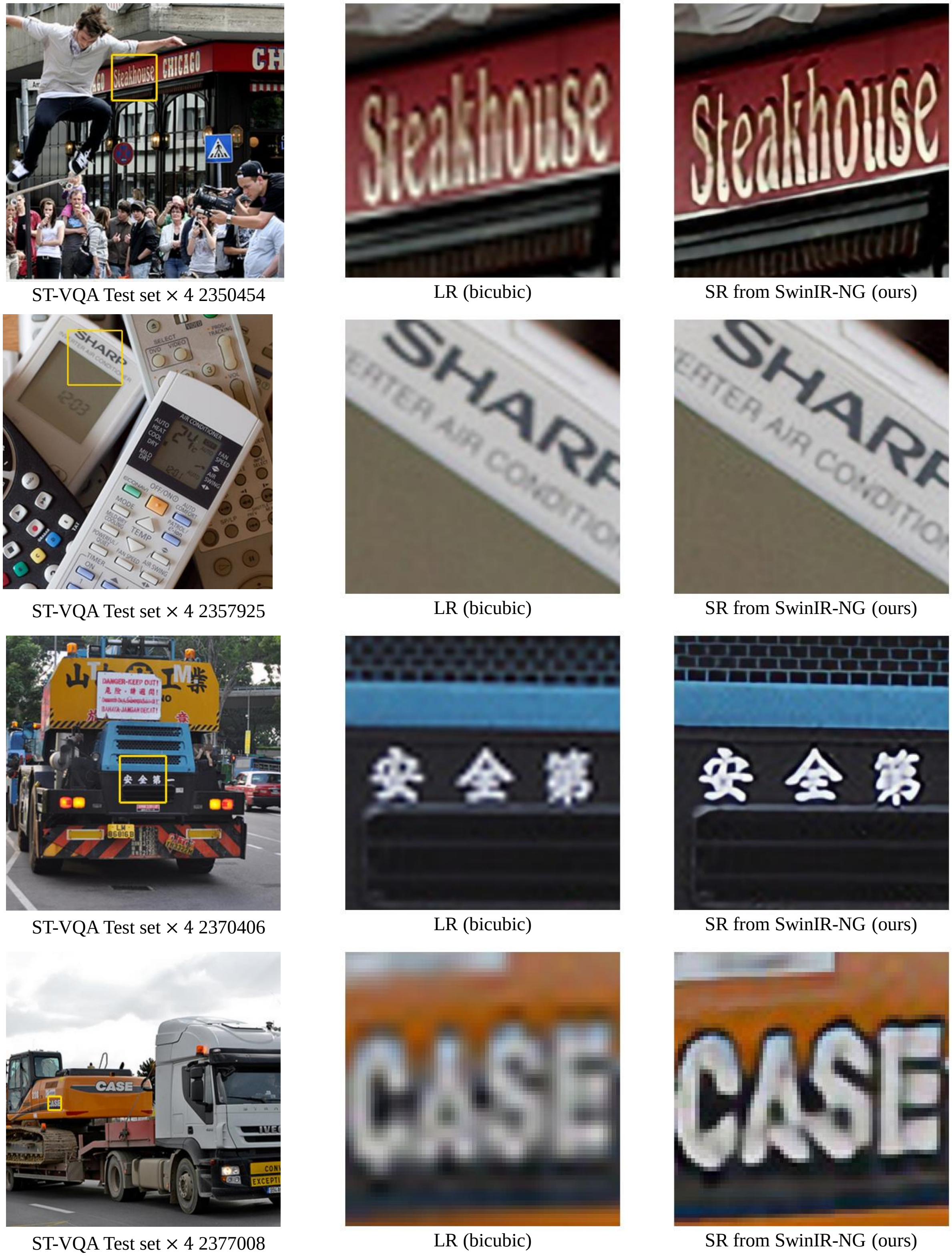


Figure L. Visual results with SwinIR-NG on CIFAR10 [18] (×4). The explanations are in Fig. K.



ST-VQA Test set × 4 2350454

LR (bicubic)

SR from SwinIR-NG (ours)

ST-VQA Test set × 4 2357925

LR (bicubic)

SR from SwinIR-NG (ours)

ST-VQA Test set × 4 2370406

LR (bicubic)

SR from SwinIR-NG (ours)

ST-VQA Test set × 4 2377008

LR (bicubic)

SR from SwinIR-NG (ours)

Figure M. SR results with SwinIR-NG on ST-VQA Test set [4] (×4). Our SwinIR-NG makes the scene text more accurate to be detected than the original LR images. We expect our work to boost detection task as well. Like Fig. K, as this figure is secondary provision, we do not compare ours with other models.

References

[1] Eirikur Agustsson and Radu Timofte. Ntire 2017 challenge on single image super-resolution: Dataset and study. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pages 126–135, 2017. 2

[2] Namhyuk Ahn, Byungkon Kang, and Kyung-Ah Sohn. Fast, accurate, and lightweight super-resolution with cascading residual network. In *Proceedings of the European conference on computer vision (ECCV)*, pages 252–268, 2018. 1

[3] Hangbo Bao, Li Dong, and Furu Wei. Beit: Bert pre-training of image transformers. *arXiv preprint arXiv:2106.08254*, 2021. 3

[4] Ali Furkan Biten, Ruben Tito, Andres Mafla, Lluis Gomez, Marçal Rusinol, Ernest Valveny, CV Jawahar, and Dimosthenis Karatzas. Scene text visual question answering. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 4291–4301, 2019. 4, 14

[5] Hanting Chen, Yunhe Wang, Tianyu Guo, Chang Xu, Yiping Deng, Zhenhua Liu, Siwei Ma, Chunjing Xu, Chao Xu, and Wen Gao. Pre-trained image processing transformer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12299–12310, 2021. 4

[6] Liangyu Chen, Xiaojie Chu, Xiangyu Zhang, and Jian Sun. Simple baselines for image restoration. In *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part VII*, pages 17–33. Springer, 2022. 4

[7] Kevin Clark, Minh-Thang Luong, Quoc V Le, and Christopher D Manning. Electra: Pre-training text encoders as discriminators rather than generators. *arXiv preprint arXiv:2003.10555*, 2020. 2, 3

[8] Marcos V Conde, Ui-Jin Choi, Maxime Burchi, and Radu Timofte. Swin2sr: Swinv2 transformer for compressed image super-resolution and restoration. *arXiv preprint arXiv:2209.11345*, 2022. 4

[9] Xiaoyi Dong, Jianmin Bao, Dongdong Chen, Weiming Zhang, Nenghai Yu, Lu Yuan, Dong Chen, and Baining Guo. Cswin transformer: A general vision transformer backbone with cross-shaped windows. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12124–12134, 2022. 4

[10] Priya Goyal, Piotr Dollár, Ross Girshick, Pieter Noordhuis, Lukasz Wesolowski, Aapo Kyrola, Andrew Tulloch, Yangqing Jia, and Kaiming He. Accurate, large minibatch sgd: Training imagenet in 1 hour. *arXiv preprint arXiv:1706.02677*, 2017. 2

[11] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16000–16009, 2022. 2, 3

[12] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 1

[13] Geoffrey E Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*, 2012. 2, 3

[14] Jia-Bin Huang, Abhishek Singh, and Narendra Ahuja. Single image super-resolution from transformed self-exemplars. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5197–5206, 2015. 1

[15] Jiwon Kim, Jung Kwon Lee, and Kyoung Mu Lee. Accurate image super-resolution using very deep convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1646–1654, 2016. 1

[16] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 2

[17] Xiangtao Kong, Xina Liu, Jinjin Gu, Yu Qiao, and Chao Dong. Reflash dropout in image super-resolution. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6002–6012, 2022. 3

[18] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009. 4, 12, 13

[19] Gustav Larsson, Michael Maire, and Gregory Shakhnarovich. Fractalnet: Ultra-deep neural networks without residuals. *arXiv preprint arXiv:1605.07648*, 2016. 2, 3

[20] Jingyun Liang, Jiezhang Cao, Guolei Sun, Kai Zhang, Luc Van Gool, and Radu Timofte. Swinir: Image restoration using swin transformer. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1833–1844, 2021. 1, 2

[21] Bee Lim, Sanghyun Son, Heewon Kim, Seungjun Nah, and Kyoung Mu Lee. Enhanced deep residual networks for single image super-resolution. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pages 136–144, 2017. 1, 2

[22] Zudi Lin, Prateek Garg, Atmadeep Banerjee, Salma Abdel Magid, Deqing Sun, Yulun Zhang, Luc Van Gool, Donglai Wei, and Hanspeter Pfister. Revisiting rcan: Improved training for image super-resolution. *arXiv preprint arXiv:2201.11279*, 2022. 2, 3

[23] Anran Liu, Yihao Liu, Jinjin Gu, Yu Qiao, and Chao Dong. Blind image super-resolution: A survey and beyond. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022. 4

[24] Ze Liu, Han Hu, Yutong Lin, Zhuliang Yao, Zhenda Xie, Yixuan Wei, Jia Ning, Yue Cao, Zheng Zhang, Li Dong, et al. Swin transformer v2: Scaling up capacity and resolution. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12009–12019, 2022. 2, 3, 4

[25] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10012–10022, 2021. 2

[26] Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*, 2016. 2

[27] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017. 2, 3

[28] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. Understanding the exploding gradient problem. *CoRR, abs/1211.5063*, 2(417):1, 2012. 2, 3

[29] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks. In *International conference on machine learning*, pages 1310–1318. PMLR, 2013. 2, 3

[30] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019. 2

[31] Wenzhe Shi, Jose Caballero, Ferenc Huszár, Johannes Totz, Andrew P Aitken, Rob Bishop, Daniel Rueckert, and Zehan Wang. Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1874–1883, 2016. 2

[32] Radu Timofte, Eirikur Agustsson, Luc Van Gool, Ming-Hsuan Yang, and Lei Zhang. Ntire 2017 challenge on single image super-resolution: Methods and results. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pages 114–125, 2017. 4

[33] Zhendong Wang, Xiaodong Cun, Jianmin Bao, Wengang Zhou, Jianzhuang Liu, and Houqiang Li. Uformer: A general u-shaped transformer for image restoration. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 17683–17693, 2022. 4

[34] Syed Waqas Zamir, Aditya Arora, Salman Khan, Munawar Hayat, Fahad Shahbaz Khan, and Ming-Hsuan Yang. Restormer: Efficient transformer for high-resolution image restoration. In *CVPR*, 2022. 4

[35] Jiale Zhang, Yulun Zhang, Jinjin Gu, Yongbing Zhang, Linghe Kong, and Xin Yuan. Accurate image restoration with attention retractable transformer. *arXiv preprint arXiv:2210.01427*, 2022. 4

[36] Xindong Zhang, Hui Zeng, Shi Guo, and Lei Zhang. Efficient long-range attention network for image super-resolution. *arXiv preprint arXiv:2203.06697*, 2022. 2, 4

[37] Yulun Zhang, Kunpeng Li, Kai Li, Lichen Wang, Bineng Zhong, and Yun Fu. Image super-resolution using very deep residual channel attention networks. In *Proceedings of the European conference on computer vision (ECCV)*, pages 286–301, 2018. 4

[38] Chen Zheng, Yulun Zhang, Jinjin Gu, Yongbing Zhang, Linghe Kong, and Xin Yuan. Cross aggregation transformer for image restoration. *arXiv preprint arXiv:2211.13654*, 2022. 4