
LMLT: Low-to-high Multi-Level Vision Transformer for Image Super-Resolution

Jeongsoo Kim, Jongho Nang, Junsuk Choe*

Sogang University

{jskim1, jhnang, jschoe}@sogang.ac.kr

Abstract

Recent Vision Transformer (ViT)-based methods for Image Super-Resolution have demonstrated impressive performance. However, they suffer from significant complexity, resulting in high inference times and memory usage. Additionally, ViT models using Window Self-Attention (WSA) face challenges in processing regions outside their windows. To address these issues, we propose the Low-to-high Multi-Level Transformer (LMLT), which employs attention with varying feature sizes for each head. LMLT divides image features along the channel dimension, gradually reduces spatial size for lower heads, and applies self-attention to each head. This approach effectively captures both local and global information. By integrating the results from lower heads into higher heads, LMLT overcomes the window boundary issues in self-attention. Extensive experiments show that our model significantly reduces inference time and GPU memory usage while maintaining or even surpassing the performance of state-of-the-art ViT-based Image Super-Resolution methods. Our codes are available at <https://github.com/jwgdmkj/LMLT>.

1 Introduction

Single Image Super-Resolution (SISR) is a technique that converts low-resolution images into high-resolution ones and has been actively researched in the field of computer vision. Traditional methods, such as nearest neighbor interpolation and bilinear interpolation, were used in the past, but recent super-resolution research has seen significant performance improvements, particularly through CNN-based methods [10, 63, 26] and Vision Transformer (ViT)-based methods [32, 64, 8].

Since the introduction of SRCNN [10], CNN-based image super-resolution architectures have advanced by utilizing multiple convolutional layers to understand contexts at various scales. These architectures deliver this understanding through residual and/or dense connections [51, 66, 60, 48, 29].

However, super-resolution using CNNs faces several issues in terms of performance and efficiency. Firstly, CNN-based models can become excessively complex and deep to improve performance, leading to increased model size and memory usage [46, 63, 62]. To mitigate this, several models share parameters between modules [1, 48], but this approach does not guarantee efficiency during inference [47]. SAFMN [47] addresses the balance between accuracy and complexity by partitioning image features using a multi-head approach [52] and implementing non-local feature relationships at various scales. However, it struggles to capture long-range dependencies due to limited kernel sizes.

ViT-based models have shown superior performance compared to CNN-based models by effectively modeling global context interactions [8, 64]. For example, SwinIR [32] utilized the Swin Transformer [35] for image super-resolution, demonstrating the effectiveness of the transformer

*Corresponding author

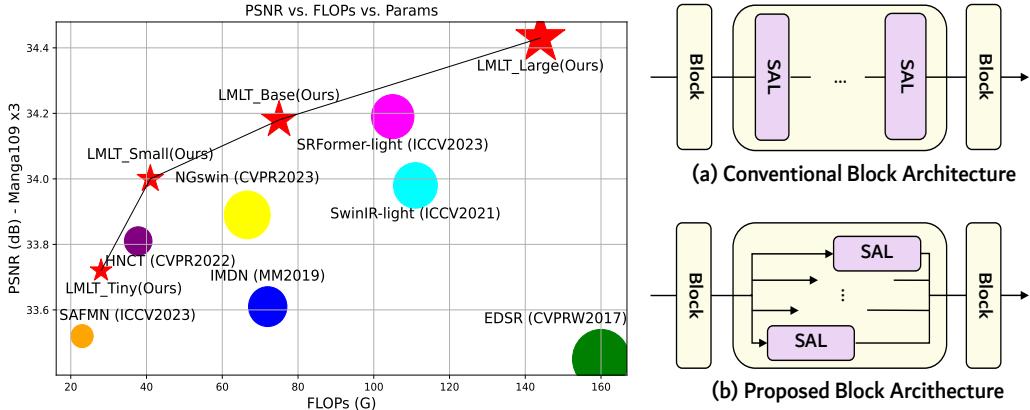


Figure 1: **Left** PSNR comparison of our proposed LMLT and other state-of-the-art models when upscaling Manga109 by 3 times. The size of each circle represents the number of parameters. Our model achieves comparable performance in terms of FLOPs when the channels are set to 36, 36 with 12 blocks, 60, and 84. **Right** (a) The conventional Self-Attention block stacks multiple Self-Attention layers in series. (b) Our proposed Self-Attention block stacks the layers in parallel. Here, SAL stands for Self Attention Layer.

architecture. Subsequently, hybrid models combining ViT and CNN have been proposed, achieving significant performance increases [15].

However, ViT models face quadratically increasing computational costs as input size grows [22, 61, 43]. To address this, Window Self-Attention (WSA) has been developed, which perform self-attention by dividing the image into windows [35]. Despite this, WSA suffers from quality degradation at window boundaries and lacks interaction between windows [17, 43]. Additionally, conventional ViT-based models stack self-attention layers in series [32, 69], which significantly increases computational load and inference time.

In this paper, we propose LMLT (Low-to-high Multi-Level Transformer) to improve efficiency during inference while maintaining performance. Similar to SAFMN, our approach uses a multi-head method [52] to split image features and apply pooling to each feature. Each head applies the self-attention mechanism. Unlike conventional self-attention blocks, which stack self-attention layers in series (Figure 1(a)), we stack them in parallel to reduce computation (Figure 1(b)). This means we integrate the number of heads and layers (depth) into a single mechanism. Note that we call the head with the most pooling the lower head, and the number of pooling applications decreases as we move to the upper heads.

Since the window size is the same for all heads, the upper heads focus on smaller areas and effectively capture local context. In contrast, the lower heads focus on larger areas and learn more global information. This approach allows us to dynamically capture both local and global information. Additionally, we introduce a residual connection [18] to pass global information from the more pooled lower heads to the less pooled upper heads. This enables the windows of the upper heads to view a wider area, thereby resolving the cross-window communication problem.

Trained on DIV2K [50] and Flickr2K [33], our extensive experiments demonstrate that ViT-based models can effectively achieve a balance between model complexity and accuracy. Compared to other state-of-the-art results, our approach significantly reduces memory usage and inference time while enhancing performance. Specifically, our base model with 60 channels and large model with 84 channels decrease memory usage to 38% and 54%, respectively, and inference time to 22% and 19% compared to ViT-based super-resolution models like NGswin [8] and SwinIR-light[32] at scale $\times 4$ scale. Moreover, our models achieve an average performance increase of 0.076db and 0.152db across all benchmark datasets.

2 Related Works

CNN-Based Image Super-Resolution is one of the most popular deep learning-based methods for enhancing image resolution. Since SRCNN [10] introduced a method to restore high-resolution (HR) images using three end-to-end layers, advancements like VDSR [26] and DRCN [27] have leveraged deeper neural network structures. These methods introduced recursive neural network structures to produce higher quality results. ESPCN [44] significantly improved the speed of super-resolution by replacing bicubic-filter upsampling with sub-pixel convolution, a technique adopted in several subsequent works [32, 69, 15]. To address the limited receptive field of CNNs, some researchers incorporated attention mechanisms into super-resolution models to capture larger areas. RCAN [65] applied channel attention to adaptively readjust the features of each channel, while SAN [9] used a second-order attention mechanism to capture more long-distance spatial contextual information. CSFM [21] dynamically modulated channel-wise and spatial attention, allowing the model to selectively emphasize various global and local features of the image. We use the same window size, similar to CNN kernels, but vary the spatial size of each feature. This allows our model to dynamically capture both local and global information by obtaining global information from smaller spatial sizes and local information from larger spatial sizes.

ViT-Based Image Super-Resolution has surpassed the performance of CNN-based models by efficiently modeling long-range dependencies and capturing global interactions between contexts [32, 46]. After the success of ViT [13] in various fields such as classification [35, 12], object detection [4, 57], and semantic segmentation [56, 45], several models have aimed to use it for low-level vision tasks. IPT [5] constructed a Transformer-based large-scale pre-trained model for image processing. However, the complexity of ViT grows quadratically with input size. To mitigate this, many approaches have aimed to reduce computational load while capturing both local and global information. For example, SwinIR [32] used the Swin-Transformer [35] model for image reconstruction. Restormer [58] organized self-attention in the channel direction to maintain global information and achieve high performance in image denoising. HAT [6] combined self-attention, which captures representative information, with channel attention, which holds global information. To combine local and global information without adding extra complexity, we add features from lower heads, which contain global information, to upper heads, which contain local information. This enables the windows to see beyond their own area and cover a larger region.

Efficient Image Super-Resolution research focuses on making super-resolution models more efficient. The CNN-based model FSRCNN [11] improved on SRCNN [10] by removing the bicubic interpolation pre-processing and increasing the scale through deconvolution, greatly speeding up computation. CARN [1] reused features at various stages through cascading residual blocks connected in a multi-stage manner. IMDN [24] progressively refined features passing through the network. However, improving performance often requires stacking many convolution layers, leading to increased computational load and memory usage.

In contrast, the ViT-based model ELAN [64] aimed to enhance spatial adaptability by using various window sizes in self-attention. HNCT [15] integrated CNN and Transformer structures to extract local features with global dependencies. NGswin [8] addressed the cross-window communication problem of the original Swin Transformer [35] by applying Ngram [40]. Despite these advances, the considerable computational load of overly deep stacked self-attention mechanisms still constrains the efficiency of ViT-based super-resolution models. To address computational load, we connect self-attention layers in parallel, integrating multi-head and depth (number of layers) to lighten the computation. This, along with reducing the spatial size of features, makes our model more efficient.

Additionally, efforts to lighten networks through methods such as knowledge distillation [19], model quantization [25], or pruning [67, 53] have been made. Some approaches differentiate between classical and lightweight image super-resolution models by using the same architecture but varying hyperparameters, such as the number of network blocks or feature channels [69, 7, 31].

3 Proposed Method

Overall Architecture (Figure 2(a)). First, we use a 3×3 convolution to extract shallow-level features from the image. Next, we stack multiple LHS Blocks (Low-to-High Self-attention Blocks) to extract deep-level features. In each LHS Block, the features go through Layer Normalization

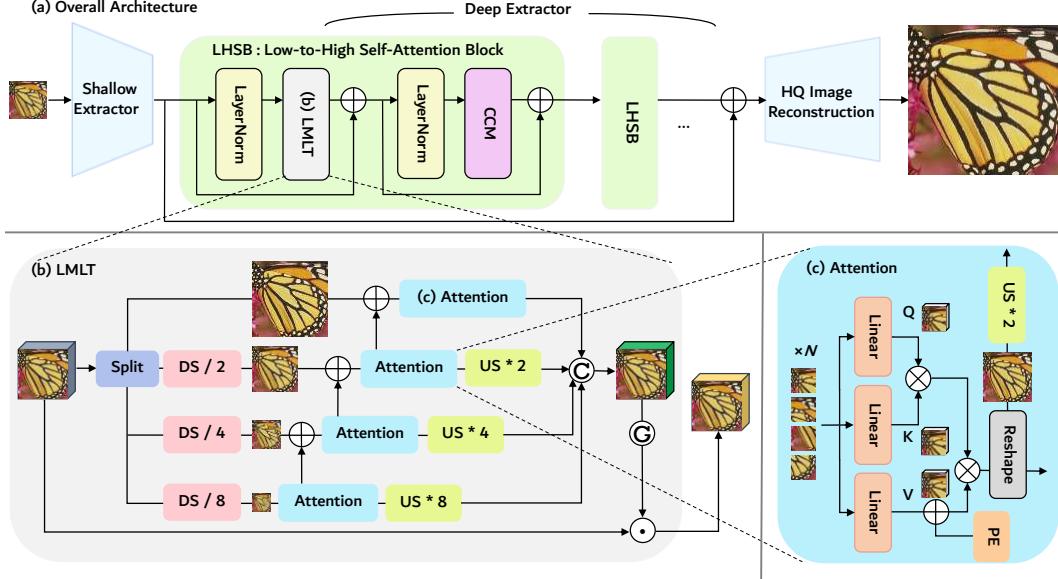


Figure 2: The architecture overview of the proposed method.

(LN) [2], our proposed LMLT (Low-to-high Multi-Level Transformer), LN again, and the CCM [47]. Residual connections are also employed. Finally, we use a 3×3 convolution filter and a pixel-shuffle layer [44] to reconstruct high-quality images. For more details on CCM [47], refer to Appendix E.

Low-to-high Multi-Level Transformer (Figure 2(b)). LMLT operates within the LHS Block. After features pass through the first LN [2], we divide them into H heads using a Multi-Head approach [52] and pool each split feature to a specified size. Specifically, the feature for the uppermost head is not pooled, and as we move to lower heads, the pooling becomes stronger, with the height and width halved for each subsequent head. Each feature then undergoes a self-attention mechanism. The output of the self-attention is interpolated to the size of the upper head’s feature and added element-wise (called a low-to-high connection). The upper head then undergoes the self-attention process again, continuing up to the topmost head. Finally, the self-attention outputs of all heads are restored to their original size, concatenated, and merged through a 1×1 convolution before being multiplied with the original feature.

Attention Layer (Figure 2(c)). In each attention layer, the feature is divided into N non-overlapping windows. The dot product of the query and key is calculated, followed by the dot product with the value. LePE [12] is used as the Positional Encoding and added to the value. The output is upscaled by a factor of 2 and sequentially passed to the upper head until it reaches the topmost head.

How the Proposed Method Works? Our proposed LMLT effectively captures both local and global regions. As seen in Figure 3, even if the window size is the same, the viewing area changes with different spatial sizes of the feature. Specifically, when self-attention is applied to smaller features, global information can be obtained. As the spatial size increases, the red window in the larger feature can utilize information beyond its own limits for self-attention calculation because it has already acquired information from other regions in the previous stage. This combination of lower heads capturing global context and upper heads capturing local context secures cross-window communication. Figure 4 visualizes the type of information each head captures. From Figure 4(a) to 4(d), the features extracted from each head when H is assumed to be 4 are visualized by averaging them along the channel dimension. The first head (4(a)) captures relatively local patterns, while fourth head (4(d)) captures global patterns. In Figure 4(e), these local and global patterns are combined to provide a comprehensive representation. By merging this with the original feature (4(f)), it emphasizes the parts that are important for super-resolution.

Computational Complexity. We improve the model’s efficiency by connecting self-attention layers in parallel and reducing spatial size. In the proposed model, given a feature $F \in \mathbb{R}^{H \times W \times D}$ and a fixed window size of $M \times M$, the number of windows in our LMLT is reduced by one-fourth as we move to lower heads, by halving the spatial size of the feature map. Additionally, since each head

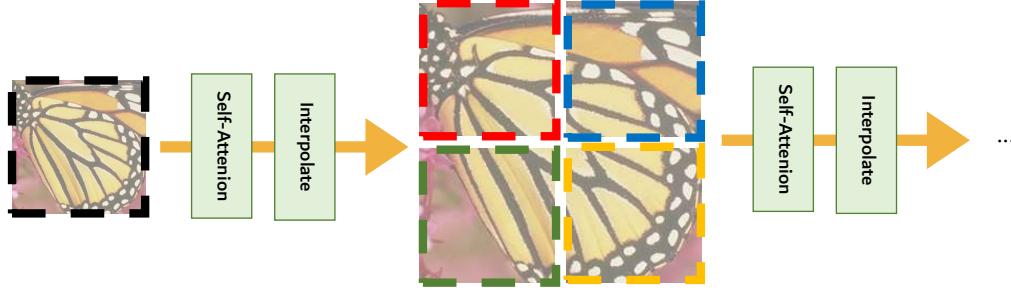


Figure 3: Self-attention(SA) at different spatial resolutions of the image with the same window size.

replaces depth, the channel count is also reduced to $\frac{D}{\text{head}}$. Therefore, the total computation for each head is given by Equation 1. Here, i means $i - 1$ th head.

$$\Omega(\text{LMLT}) = 4 \left[\frac{hw}{4^i} \left(\frac{D}{\text{head}} \right)^2 \right] + 2 \left[\frac{M^2 hw}{4^i} \frac{D}{\text{head}} \right]. \quad (1)$$

On the other hand, in WSA [35, 32], the self-attention layers are stacked in series, and the spatial size and channel of the feature do not decrease, so the number of windows remains $\frac{HW}{M^2}$, and the channel count stays at D . Therefore, the total computation amount is shown in Equation 2.

$$\Omega(\text{WSA}) = 4hwD^2 + 2M^2hwD. \quad (2)$$

Therefore, in our proposed model, if the number of heads is greater than 1, both the number of windows and channels decrease compared to WSA [35, 32], resulting in reduced computational load. The more heads there are, the greater the reduction in computational load.

4 Experiments

Datasets. Following previous studies [32, 46, 15], we use DIV2K [50], consisting of 800 images, and Flickr2K [33], consisting of 2,650 images, as training datasets. For testing, we use the Set5 [3], Set14 [59], BSD100 [41], Urban100 [23], and Manga109 [42] datasets.

Implementation Details. We categorize our model into four types: a Tiny model with 36 channels, a Small model with 36 channels and 12 blocks, a Base model with 60 channels, and a Large model with 84 channels. First, the low-resolution (LR) images used as training inputs are cropped into 64×64 patches. Rotation and horizontal flip augmentations are applied to this training data. The number of blocks, heads, and growth ratio are set to 8 (except for the Small model), 4, and 2, respectively. We use the Adam Optimizer [28] with $\beta_1 = 0.9$ and $\beta_2 = 0.99$, running for 500,000 iterations. The initial learning rate is set to 1×10^{-3} and is reduced to at least 1×10^{-5} using the cosine annealing scheme [36]. To accelerate the speed of our experiments, we set `backends.cudnn.benchmark` to True and `backends.cudnn.deterministic` to False for the 36-channel model. To account for potential variability in the results due to this setting, we conduct three separate experiments with the LMLT-Tiny model and reported the average of these results. All other experiments are conducted only once.

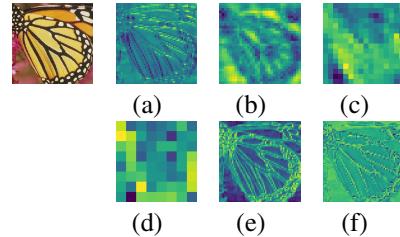


Figure 4: Features from each head ((a) to (d)), aggregated feature (e), and feature multiplied with the original feature (f).

Table 1: Comparisons with our LMLT-Base, LMLT-Large and other Super-Resolution models on multiple benchmark datasets. Best and second-best performance are in red and blue color.

Scale	Method	#Params	#FLOPs	Set5	Set14	B100	Urban100	Manga109
$\times 2$	IMDN [24]	694K	156G	38.00/0.9605	33.63/0.9177	32.19/0.8996	32.17/0.9283	38.88/0.9774
	LatticeNet [39]	756K	170G	38.06/0.9607	33.70/0.9187	32.20/0.8999	32.25/0.9288	38.94/0.9774
	RFDN-L [34]	626K	146G	38.08 /0.9606	33.67/0.9190	32.18/0.8996	32.24/0.9290	38.95/0.9773
	SRPN-Lite [67]	609K	140G	38.10 /0.9608	33.70/0.9189	32.25/0.9005	32.26/0.9294	-
	HNCT [15]	357K	82G	38.08 /0.9608	33.65/0.9182	32.22/0.9001	32.22/0.9294	38.87/0.9774
	FMEN [14]	748K	172G	38.10 /0.9609	33.75/0.9192	32.26/0.9007	32.41/0.9311	38.95/ 0.9778
	NGswin [8]	990K	140G	38.05/0.9610	33.79 /0.9199	32.27 /0.9008	32.53 /0.9324	38.97 /0.9777
	LMLT-Base(Ours)	652K	158G	38.10 /0.9610	33.76 /0.9201	32.28 /0.9012	32.52 /0.9316	39.24 /0.9783
	ESRT [37]	751K	-	38.03/0.9600	33.75/0.9184	32.25/0.9001	32.58/0.9318	39.12/0.9774
	SwinIR-light [32]	910K	244G	38.14/0.9611	33.86/0.9206	32.31/0.9012	32.76 /0.9340	39.12/0.9783
$\times 3$	ELAN [64]	621K	203G	38.17/0.9611	33.94 /0.9207	32.30/0.9012	32.76 /0.9340	39.11/0.9782
	SRformer-Light [69]	853K	236G	38.23 /0.9613	33.94 /0.9209	32.36 /0.9019	32.91 /0.9353	39.28 /0.9785
	LMLT-Large(Ours)	1,270K	306G	38.18 /0.9612	33.96 /0.9212	32.33 /0.9017	32.75/0.9336	39.41 /0.9786
	IMDN [24]	703K	72G	34.36/0.9270	30.32/0.8417	29.09/0.8046	28.17/0.8519	33.61/0.9445
	LatticeNet [39]	765K	76G	34.40/0.9272	30.32/0.8416	29.10/0.8049	28.19/0.8513	33.63/0.9442
	RFDN-L [34]	633K	66G	34.47/0.9280	30.35/0.8421	29.11/0.8053	28.32/0.8547	33.78/0.9458
	SRPN-Lite [67]	615K	63G	34.47/0.9280	30.38/0.8425	29.16/0.8061	28.22/0.8534	-
	HNCT [15]	363K	38G	34.47/0.9275	30.44 /0.8439	29.15/0.8067	28.28/0.8557	33.81/0.9459
	FMEN [14]	757K	77G	34.45/0.9275	30.40/0.8435	29.17/0.8063	28.33/0.8562	33.86/0.9462
	NGswin [8]	1,007K	67G	34.52 /0.9282	30.53 /0.8456	29.19 /0.8078	28.52 /0.8603	33.89 /0.9470
$\times 4$	LMLT-Base(Ours)	660K	75G	34.58 /0.9285	30.53 /0.8458	29.21 /0.8084	28.48 /0.8581	34.18 /0.9477
	ESRT [37]	751K	-	34.42/0.9268	30.43/0.8433	29.15/0.8063	28.46/0.8574	33.95/0.9455
	SwinIR-light [32]	918K	111G	34.62/0.9289	30.54/0.8463	29.20/0.8082	28.66/0.8624	33.98/0.9478
	ELAN [64]	629K	90G	34.61/0.9288	30.55/0.8463	29.21 /0.8081	28.69/0.8624	34.00/0.9478
	SRformer-Light [69]	861K	105G	34.67 /0.9296	30.57 /0.8469	29.26 /0.8099	28.81 /0.8655	34.19 /0.9489
	LMLT-Large(Ours)	1,279K	144G	34.64 /0.9293	30.60 /0.8471	29.26 /0.8097	28.72 /0.8626	34.43 /0.9491
	IMDN [24]	715K	41G	32.21/0.8948	28.58/0.7811	27.56/0.7353	26.04/0.7838	30.45/0.9075
	LatticeNet [39]	777K	44G	32.18/0.8943	28.61/0.7812	27.57/0.7355	26.14/0.7844	30.54/0.9075
	RFDN-L [34]	643K	38G	32.28/0.8957	28.61/0.7818	27.58/0.7363	26.20/0.7883	30.61/0.9096
	SRPN-Lite [67]	623K	36G	32.24/0.8958	28.69/0.7836	27.63/0.7373	26.16/0.7875	-
$\times 4$	HNCT [15]	373K	22G	32.31/0.8957	28.71/0.7834	27.63/0.7381	26.20/0.7896	30.70/0.9112
	FMEN [14]	769K	44G	32.24/0.8955	28.70 /0.7839	27.63/0.7379	26.28/0.7908	30.70/0.9107
	NGswin [8]	1,019K	36G	32.33 /0.8963	28.78 /0.7859	27.66 /0.7396	26.45 /0.7963	30.80 /0.9128
	LMLT-Base(Ours)	672K	41G	32.38 /0.8971	28.79 /0.7859	27.70 /0.7403	26.44 /0.7947	31.09 /0.9139
	ESRT [37]	751K	-	32.19/0.8947	28.69/0.7833	27.69/0.7379	26.39/0.7962	30.75/0.9100
	SwinIR-light [32]	930K	64G	32.44/0.8976	28.77/0.7858	27.69/0.7406	26.47/0.7980	30.92/0.9151
	ELAN [64]	640K	54G	32.43/0.8975	28.78/0.7858	27.69/0.7406	26.54/0.7982	30.92/0.9150
	SRformer-Light [69]	873K	63G	32.51 /0.8988	28.82 /0.7872	27.73 /0.7422	26.67 /0.8032	31.17 /0.9165
	LMLT-Large(Ours)	1,295K	78G	32.48 /0.8987	28.87 /0.7879	27.75 /0.7421	26.63 /0.8001	31.32 /0.9163

Evaluation Metrics. The quality of the recovered high-resolution images is evaluated using Peak Signal-to-Noise Ratio (PSNR) and Structural Similarity Index (SSIM) [55]. These metrics are calculated on the Y channel of the YCbCr color space. To test the efficiency of our model, we follow the method of SAFMN [47], measuring GPU memory consumption (#GPU Mem) and inference time (#AVG Time) for scaling a total of 50 images across various models. #GPU Mem, obtained through PyTorch’s `torch.cuda.max_memory_allocated()`, represents the maximum memory consumption during inference, and #AVG Time is the average time per image for inferring a total of 50 LR images at $\times 2$, $\times 3$, and $\times 4$ scales. The results for $\times 2$, $\times 3$, and $\times 4$ scaling are based on upscaling random images of sizes 640×360 , 427×240 , and 320×180 , respectively.

4.1 Comparisons with State-of-the-Art Methods

Image Reconstruction Comparisons. To evaluate the performance of the proposed model, we compare our models with other state-of-the-art efficient and lightweight SR models at different scaling factors. PSNR, SSIM [55], the number of parameters, and FLOPs are used as the main performance evaluation metrics. Note that FLOPs refer to the computational amount required to create an image with a resolution of 1280×720 .

We first compare the LMLT-Base with IMDN [24], LatticeNet [39], RFDN-L [34], SRPN-Lite [67], HNCT [15], FMEN [14], and NGswin [8]. Table 1 shows that our LMLT-Base achieves the best or second-best performance on most benchmark datasets. Notably, we observe a significant performance increase on the Manga109 dataset, while our model uses up to 30% fewer parameters compared to

Table 2: The memory consumption and inference times are reported. A single RTX 3090 GPU is used.

Scale	Method	#GPU Mem [M]	#Avg Time [ms]	Set5	Set14	B100	Urban100	Manga109
$\times 2$	LMLT-Tiny(Ours)	324.01	57.37	38.01/0.9606	33.59/0.9183	32.19/0.8999	32.04/0.9273	38.90/0.9775
	LMLT-Small(Ours)	324.5	84.22	38.05/0.9608	33.65/0.9187	32.24/0.9006	32.31/0.9298	39.10/0.9780
	HNCT [15]	1200.55	351.49	38.08/0.9608	33.65/0.9182	32.22/0.9001	32.22/0.9294	38.87/0.9774
	NGswin [8]	1440.40	375.19	38.05/0.9610	33.79/0.9199	32.27/0.9008	32.53/0.9324	38.97/0.9777
	LMLT-Base(Ours)	567.75	81.64	38.10/0.9610	33.76/0.9201	32.28/0.9012	32.52/0.9316	39.24/0.9783
	SwinIR-light [32]	1278.64	944.11	38.14/0.9611	33.86/0.9206	32.31/0.9012	32.76/0.9340	39.12/0.9783
$\times 3$	SRformer-Light [69]	1176.15	1006.48	38.23/0.9613	33.94/0.9209	32.36/0.9019	32.91/0.9353	39.28/0.9785
	LMLT-Large(Ours)	717.31	123.07	38.18/0.9612	33.96/0.9212	32.33/0.9017	32.75/0.9336	39.41/0.9786
	LMLT-Tiny(Ours)	151.96	31.06	34.36/0.9271	30.37/0.8427	29.12/0.8057	28.10/0.8503	33.72/0.9448
	LMLT-Small(Ours)	152.5	44.22	34.50/0.9280	30.47/0.8446	29.16/0.8070	28.29/0.8544	33.99/0.9464
	HNCT [15]	545.64	117.20	34.47/0.9275	30.44/0.8439	29.15/0.8067	28.28/0.8557	33.81/0.9459
	NGswin [8]	696.97	168.49	34.52/0.9282	30.53/0.8456	29.19/0.8078	28.52/0.8603	33.89/0.9470
$\times 4$	LMLT-Base(Ours)	266.31	41.43	34.58/0.9285	30.53/0.8458	29.21/0.8084	28.48/0.8581	34.18/0.9477
	SwinIR-light [32]	587.63	287.96	34.62/0.9289	30.54/0.8463	29.20/0.8082	28.66/0.8624	33.98/0.9478
	SRformer-Light [69]	529.28	312.37	34.67/0.9296	30.57/0.8469	29.26/0.8099	28.81/0.8655	34.19/0.9489
	LMLT-Large(Ours)	338.36	58.68	34.64/0.9293	30.60/0.8471	29.26/0.8097	28.72/0.8626	34.43/0.9491
	LMLT-Tiny(Ours)	81.44	23.54	32.19/0.8947	28.64/0.7823	27.60/0.7369	26.08/0.7838	30.60/0.9083
	LMLT-Small(Ours)	81.92	31.01	32.31/0.8968	28.74/0.7846	27.66/0.7387	26.26/0.7894	30.87/0.9117

the next highest performing model, NGswin [8], while the PSNR increases by 0.27dB, 0.29dB, and 0.29dB, respectively, at all scales.

Next, we compare the LMLT-Large model with other SR models. The comparison group includes ESRT [37], SwinIR-light [32], ELAN [64], and SRFormer-light [69]. As shown in Table 1, our large model ranks first or second in performance on most datasets. Among the five test datasets, LMLT-Large shows the best performance for all scales on the Manga109 [42] dataset compared to others, and also the best performance on the Set14 [59] dataset. Specifically, compared to SRFormer-light [69], which showed the highest performance on Urban100 [23] among the comparison group, our model shows performance gains of 0.13dB, 0.24dB, and 0.15dB at each scale on the Manga109 [42] dataset. In addition to this, we demonstrate that our model has a significant advantage in inference time and GPU memory occupancy at next paragraph. The comparison results of LMLT-Tiny and LMLT-Small with other state-of-the-art models can be found in Appendix F.

Memory and Running Time Comparisons. To test the efficiency of the proposed model, we compare the performance of our LMLT model against other ViT-based state-of-the-art super-resolution models at different scales. We evaluate LMLT-Base against NGswin [8] and HNCT [15], and LMLT-Large against SwinIR-light [32] and SRFormer-light [69]. The results are shown in Table 2.

We observe that LMLT-Base and LMLT-Large is quite efficient in terms of inference speed and memory usage compared to other ViT-based SR models. Specifically, compared to NGswin [8], our LMLT-Base maintains similar performance while reducing memory usage by 61%, 62%, and 61% for $\times 2$, $\times 3$, and $\times 4$ scales, respectively, and decreasing inference time by an average of 78%, 76%, and 78%. Similarly, when comparing SwinIR [32] and our LMLT-Large, despite maintaining similar performance, memory usage decreases by 44%, 43%, and 46% for each scale, respectively, and inference time decreases by an average of 87%, 80%, and 81%. This demonstrates both the efficiency and effectiveness of the proposed model.

Table 3 shows the time consumption for module in the proposed method and SwinIR-light [32], specifically detailing the time from the first Layer Normalization (LN) [2] to the LHSB and WSA. Our LMLT significantly reduces the time required for the self-attention mechanism. Especially, LMLT-Large achieves time reductions of 94% at the $\times 2$ scale, 90% at the $\times 3$ scale, and 88% at the $\times 4$ scale compared to SwinIR. Given the similar performance between SwinIR

Table 3: Time consumption from LN to LHSB (Ours) and WSA (SwinIR [32]). A RTX 3090 GPU is used.

method	$\times 2$	$\times 3$	$\times 4$
LMLT-Tiny(Ours)	35.28ms	23.21ms	18.36ms
LMLT-Base(Ours)	49.44ms	25.51ms	21.18ms
LMLT-Large(Ours)	68.97ms	32.72ms	22.66ms
SwinIR-Light [32]	1084.57ms	336.00ms	185.23ms

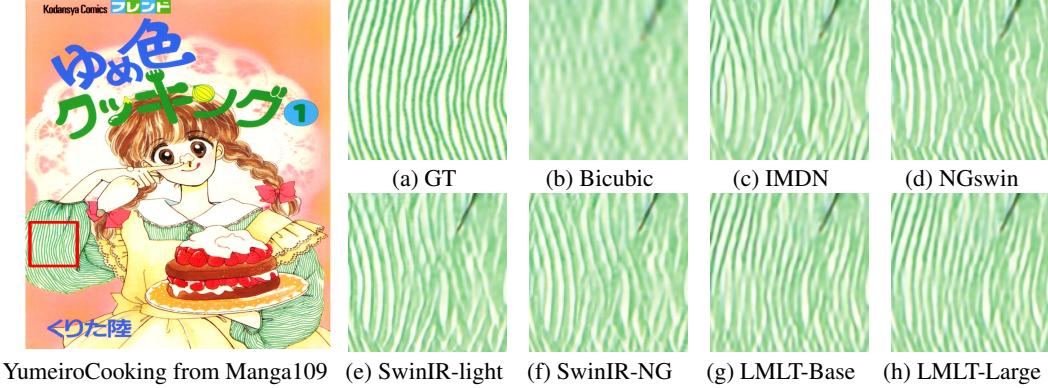


Figure 5: Visual comparisons for $\times 4$ SR on Manga109 dataset. Compared with the results in (c) to (f), the Ours(LMLT-Base(g), LMLT-Large(h)) restore much more accurate and clear images. More results are in the Appendix F.

Table 4: Performance results when the low-to-high element-wise sum removed. Better results are highlighted.

Scale	Method	Set5	Set14	B100	Urban100	Manga109
$\times 2$	LMLT-Tiny	38.01 /0.9606	33.59 / 0.9183	32.19 /0.8999	32.04 / 0.9273	38.90 /0.9775
	LMLT-Tiny <i>w/o</i> sum	38.00/0.9606	33.58/0.9181	32.18/0.8999	31.99/0.9268	38.88/0.9775
$\times 3$	LMLT-Tiny	34.36/0.9271	30.37 / 0.8427	29.12 / 0.8057	28.10 / 0.8503	33.72 /0.9448
	LMLT-Tiny <i>w/o</i> sum	34.40 / 0.9272	30.35/0.8425	29.11/0.8056	28.05/0.8496	33.71/0.9448
$\times 4$	LMLT-Tiny	32.19 / 0.8947	28.64/0.7823	27.60/ 0.7369	26.08 / 0.7838	30.60/ 0.9083
	LMLT-Tiny <i>w/o</i> sum	32.16/0.8944	28.64/0.7823	27.60/0.7368	26.04/0.7827	30.64 /0.9078

and our method, this represents a significant increase in efficiency. Note that the inference time for a total of 50 random images is measured. The time measurements are conducted using `torch.cuda.Event`'s record and `elapsed_time()`, and due to hardware access latency during log output, the time of the modules might be longer than the time report in Table 2. Tables comparing the memory usage and inference speed of our LMLT with other models can be found in Appendix F.

Qualitative Comparisons. Figure 5 illustrates the differences on the Manga109 [42] dataset between our model and other models. As shown, our LMLT successfully reconstructs areas with continuous stripes better than other models. Additionally, we include more comparison images, and further compare our proposed models LMLT-Tiny with CARN [1], EDSR [33], PAN [68], ShuffleMixer [46] and SAFMN [47] on the Urban100 [23] dataset at $\times 4$ scale. Detailed results can be seen in Appendix F.

4.2 Ablation Study

Effects of Low-to-high Connection. We examine the effects of the low-to-high element-wise sum (low-to-high connection) and downsizing elements of our proposed model. As shown in Table 4, the low-to-high connection yields significant results. Specifically, on the Urban100 [23] dataset, PSNR increases by 0.04 dB to 0.05 dB across all scales, and SSIM [55] increases by nearly 0.0011 at the $\times 4$ scale, demonstrating the benefits of including the low-to-high connection. Appendix B visualizes the differences in features on Urban100 [23] with and without the low-to-high connection, showing that it significantly reduces the boundary lines between windows. Additionally, experiments adding the proposed low-to-high connection to SAFMN [47] for $\times 2$, $\times 3$, and $\times 4$ scales are also provided in Appendix B.

Effects of Multi-Scale Heads. Table 5 validates the effectiveness of using multiple scales by experimenting with cases where pooling is not applied to any head. Specifically, we compare our proposed LMLT with cases where pooling and the low-to-high connection are not applied, as well as cases where merging is also not performed. The results show that performance is lower in all cases compared to the proposed model. Appendix B demonstrates that when pooling is not applied, the

Table 5: Performance with or without pooling and merging. Best results are highlighted in **bold**.

Scale	Method	#Params	#FLOPs	#Acts	Set5	Set14	B100	Urban100	Manga109
$\times 2$	LMLT-Tiny	239K	59G	603M	38.01/0.9606	33.59/0.9183	32.19/0.8999	32.04/0.9273	38.90/0.9775
	LMLT-Tiny w/o pool	239K	67G	1223M	37.98/0.9605	33.56/0.9178	32.16/0.8996	31.87/0.9255	38.79/0.9773
	LMLT-Tiny w/o pool and merge	229K	64G	1152M	37.95/0.9604	33.51/0.9173	32.14/0.8993	31.76/0.9245	38.68/0.9771
$\times 3$	LMLT-Tiny	244K	28G	283M	34.36/0.9271	30.37/0.8427	29.12/0.8057	28.10/0.8503	33.72/0.9448
	LMLT-Tiny w/o pool	244K	32G	572M	34.36/0.9270	30.34/0.8421	29.10/0.8051	28.02/0.8488	33.66/0.9445
	LMLT-Tiny w/o pool and merge	234K	31G	539M	34.28/0.9265	30.31/0.8417	29.07/0.8044	27.94/0.8467	33.55/0.9438
$\times 4$	LMLT-Tiny	251K	15G	152M	32.19/0.8947	28.64/0.7823	27.60/0.7369	26.08/0.7838	30.60/0.9083
	LMLT-Tiny w/o pool	251K	17G	308M	32.12/0.8940	28.61/0.7820	27.58/0.7362	26.01/0.7815	30.51/0.9074
	LMLT-Tiny w/o pool and merge	240K	17G	290M	32.07/0.8934	28.60/0.7817	27.56/0.7355	25.95/0.7795	30.45/0.9064

Table 6: Ablation studies on each component of our method at scale $\times 2$. LMLT-Tiny is used.

Ablation	Variants	#Param	#Flops	#GPU Mem	Set5	Set14	B100	Urban100	Manga109
Baseline	-	239K	59G	324M	38.01/0.9606	33.59/0.9183	32.19/0.8999	32.04/0.9273	38.90/0.9775
Module	LHSB \rightarrow None	214K	52G	241M	37.88/0.9601	33.39/0.9160	32.05/0.8981	31.45/0.9210	38.40/0.9765
	CCM \rightarrow None	31K	8G	323M	37.26/0.9573	32.85/0.9113	31.64/0.8927	30.21/0.9071	36.91/0.9720
	CCM \rightarrow MLP	73K	18G	324M	37.71/0.9593	33.25/0.9150	31.96/0.8968	31.19/0.9187	38.03/0.9754
	CCM \rightarrow IVF 4 Blocks	101K	19G	324M	37.91/0.9603	33.46/0.9173	32.11/0.8990	31.71/0.9243	38.62/0.9770
	No Aggregation	229K	30G	307M	37.88/0.9601	33.40/0.9166	32.06/0.8984	31.49/0.9217	38.47/0.9765
Act / Aggr	No Activation	239K	59G	324M	37.99/0.9606	33.55/0.9178	32.17/0.8997	31.94/0.9262	38.84/0.9774
	No Aggr, No Act	229K	56G	324M	37.95/0.9606	33.53/0.9175	32.15/0.8994	31.82/0.9250	38.73/0.9771
	GELU \rightarrow None	239K	59G	324M	38.03/0.9606	33.60/0.9184	32.19/0.9000	32.05/0.9272	38.91/0.9776
	No PE	236K	59G	309M	37.98/0.9606	33.55/0.9176	32.18/0.8998	31.98/0.9267	38.86/0.9774
PE	LePE \rightarrow RPE[35]	244K	59G	369M	38.02/0.9606	33.62/0.9182	32.20/0.9000	32.05/0.9275	38.90/0.9775

lack of information connection between windows hinders the proper capture of informative features, even though spatial information is retained.

Importance of LHSB, CCM, and MLP. We analyze the impact of LHSB and CCM [47] and their interplay. Following the approach in SAFMN [47], we examine performance by individually removing LHSB and CCM [47]. Results are shown in the ‘Module’ row of Table 6. Removing LHSB reduces the number of parameters by nearly 10%, decreases memory usage to nearly 74%, and drops PSNR by 0.59 dB on the Urban100 [23] dataset. Conversely, removing CCM reduces the number of parameters by nearly 90% and PSNR by 1.83 dB. Adding an MLP after the self-attention module, as done in traditional Transformers [52, 13], reduces parameters by about 69% and PSNR by approximately 0.85 dB. To maintain the same number of layers as in the previous two experiments (8 layers), we conducted another experiment with only 4 blocks, resulting in a 49% reduction in parameters and only a 0.55 dB drop in PSNR, indicating the least performance loss. This suggests that the combination of LHSB and CCM effectively extracts features. Additionally, incorporating FMBConv [49] reduces parameters by nearly 58% and PSNR by 0.28 dB, while memory usage remains similar.

Importance of Aggregation and Activation. We analyze the impact of aggregating features from each head using a 1×1 convolution or applying activation before multiplying with the original input. Results are shown in the ‘Act / Aggr’ row of Table 6. Without aggregation, PSNR decreases by 0.10 dB on the Urban100 [23] dataset. If features are directly output without applying activation and without multiplying with the original input, PSNR decreases by 0.12 dB. Omitting both steps leads to an even greater decrease of 0.22 dB, indicating that including both aggregation and activation is more efficient. Conversely, multiplying features directly to the original feature without the activation function improves performance by 0.1 dB. Detailed experimental results are discussed in Appendix C.

Importance of Positional Encoding. Lastly, we examine the role of Positional Encoding (PE) in performance improvement. Results are shown in the ‘PE’ row of Table 6. Removing PE results in decreased performance across all benchmark datasets, notably with a PSNR drop of 0.06 dB and an SSIM decrease of 0.0006 on the Urban100 [23] dataset. Using RPE [35] results in a maximum PSNR increase of 0.03 dB on the Set14 [59] dataset, but has little effect on other datasets. Additionally, parameters and GPU memory increase by 5K and 45M, respectively.

5 Conclusion

In this paper, we introduced the Low-to-high Multi-Level Transformer (LMLT) for efficient image super-resolution. By combining multi-head and depth reduction, our model addresses the excessive computational load and memory usage of traditional ViT models. In addition to this, LMLT applies self-attention to features at various scales, aggregating lower head outputs to inform higher heads, thus solving the cross-window communication issue. Our extensive experiments demonstrate that LMLT achieves a favorable balance between model complexity and performance, significantly reducing memory usage and inference time while maintaining or improving image reconstruction quality. This makes the proposed LMLT a highly efficient solution for image super resolution tasks, suitable for deployment on resource-constrained devices.

References

- [1] 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)*, September 2018.
- [2] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.
- [3] Marco Bevilacqua, Aline Roumy, Christine M. Guillemot, and Marie-Line Alberi-Morel. Low-complexity single-image super-resolution based on nonnegative neighbor embedding. In *In Proceedings of the British Machine Vision Conference (BMVC)*, 2012. URL <https://api.semanticscholar.org/CorpusID:5250573>.
- [4] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2020.
- [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 (CVPR)*, 2021.
- [6] Xiangyu Chen, Xintao Wang, Jiantao Zhou, Yu Qiao, and Chao Dong. Activating more pixels in image super-resolution transformer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 22367–22377, June 2023.
- [7] Zheng Chen, Yulun Zhang, Jinjin Gu, Linghe Kong, Xiaokang Yang, and Fisher Yu. Dual aggregation transformer for image super-resolution. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023.
- [8] Haram Choi, Jeongmin Lee, and Jihoon Yang. N-gram in swin transformers for efficient lightweight image super-resolution. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2071–2081, 2023.
- [9] Tao Dai, Jianrui Cai, Yongbing Zhang, Shu-Tao Xia, and Lei Zhang. Second-order attention network for single image super-resolution. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11057–11066, 2019. doi: 10.1109/CVPR.2019.01132.
- [10] Chao Dong, Chen Change Loy, Kaiming He, and Xiaoou Tang. Learning a deep convolutional network for image super-resolution. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2014. URL <https://api.semanticscholar.org/CorpusID:18874645>.
- [11] Chao Dong, Chen Change Loy, and Xiaoou Tang. Accelerating the super-resolution convolutional neural network. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2016.
- [12] 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 Computer Vision and Pattern Recognition (CVPR)*, pages 12114–12124, 2021. URL <https://api.semanticscholar.org/CorpusID:235694312>.
- [13] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *In Proceedings of the International Conference on Learning Representations (ICLR)*, 2021. URL <https://openreview.net/forum?id=YicbFdNTTy>.

- [14] Zongcai Du, Ding Liu, Jie Liu, Jie Tang, Gangshan Wu, and Lean Fu. Fast and memory-efficient network towards efficient image super-resolution. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 853–862, 2022.
- [15] Jinsheng Fang, Hanjiang Lin, Xinyu Chen, and Kun Zeng. A hybrid network of cnn and transformer for lightweight image super-resolution. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1103–1112, 2022.
- [16] Jinjin Gu and Chao Dong. Interpreting super-resolution networks with local attribution maps. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9199–9208, 2021.
- [17] Ali Hatamizadeh, Greg Heinrich, Hongxu Yin, Andrew Tao, Jose M Alvarez, Jan Kautz, and Pavlo Molchanov. Fastervit: Fast vision transformers with hierarchical attention. *arXiv preprint arXiv:2306.06189*, 2023.
- [18] Kaiming He, X. Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2015. URL <https://api.semanticscholar.org/CorpusID:206594692>.
- [19] Zibin He, Tao Dai, Jian Lu, Yong Jiang, and Shu-Tao Xia. Fakd: Feature-affinity based knowledge distillation for efficient image super-resolution. In *Proceedings of the International Conference on Image Processing (ICIP)*, 2020.
- [20] Dan Hendrycks and Kevin Gimpel. Gaussian error linear units. *arXiv preprint arXiv:1606.08415*, 2016.
- [21] Yanting Hu, Jie Li, Yuanfei Huang, and Xinbo Gao. Channel-wise and spatial feature modulation network for single image super-resolution. *IEEE Transactions on Circuits and Systems for Video Technology*, 30:3911–3927, 2018. URL <https://api.semanticscholar.org/CorpusID:52893847>.
- [22] Huaibo Huang, Xiaoqiang Zhou, and Ran He. Orthogonal transformer: an efficient vision transformer backbone with token orthogonalization. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, 2024.
- [23] 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 (CVPR)*, pages 5197–5206, 2015. doi: 10.1109/CVPR.2015.7299156.
- [24] Zheng Hui, Xinbo Gao, Yunchu Yang, and Xiumei Wang. Lightweight image super-resolution with information multi-distillation network. In *ACM MM*, 2019.
- [25] Andrey Ignatov, Radu Timofte, Maurizio Denna, and et al. Efficient and accurate quantized image super-resolution on Mobile NPUs, Mobile AI & AIM 2022 challenge: Report. In *Proceedings of the IEEE Conference on European Conference on Computer Vision (ECCV) Workshops*, 2022.
- [26] Jiwon Kim, Jung Kwon Lee, and Kyoung Mu Lee. Accurate image super-resolution using very deep convolutional networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.
- [27] Jiwon Kim, Jung Kwon Lee, and Kyoung Mu Lee. Deeply-recursive convolutional network for image super-resolution. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1637–1645, 2016. doi: 10.1109/CVPR.2016.181.
- [28] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015.
- [29] Christian Ledig, Lucas Theis, Ferenc Huszár, Jose Caballero, Andrew Cunningham, Alejandro Acosta, Andrew Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, et al. Photo-realistic single image super-resolution using a generative adversarial network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.
- [30] Wenbo Li, Kun Zhou, Lu Qi, Nianjuan Jiang, Jiangbo Lu, and Jiaya Jia. LAPAR: Linearly-assembled pixel-adaptive regression network for single image super-resolution and beyond. In *NeurIPS*, 2020.
- [31] Xiang Li, Jinshan Pan, Jinhui Tang, and Jiangxin Dong. Dlgsanet: Lightweight dynamic local and global self-attention networks for image super-resolution. *arXiv preprint arXiv:2301.02031*, 2023.
- [32] Jingyun Liang, Jiezhang Cao, Guolei Sun, Kai Zhang, Luc Van Gool, and Radu Timofte. Swinir: Image restoration using swin transformer. *arXiv preprint arXiv:2108.10257*, 2021.

- [33] 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 (CVPR) Workshops*, July 2017.
- [34] Jie Liu, Jie Tang, and Gangshan Wu. Residual feature distillation network for lightweight image super-resolution. In *European Conference on Computer Vision*, pages 41–55. Springer, 2020.
- [35] 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 (ICCV)*, pages 10012–10022, 2021.
- [36] Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. In *ICLR*, 2017.
- [37] Zhisheng Lu, Hong Liu, Juncheng Li, and Linlin Zhang. Efficient transformer for single image super-resolution. *arXiv:2108.11084*, 2021.
- [38] Wenjie Luo, Yujia Li, Raquel Urtasun, and Richard Zemel. Understanding the effective receptive field in deep convolutional neural networks, 2017. URL <https://arxiv.org/abs/1701.04128>.
- [39] Xiaotong Luo, Yuan Xie, Yulun Zhang, Yanyun Qu, Cuihua Li, and Yun Fu. Latticenet: Towards lightweight image super-resolution with lattice block. In *European Conference on Computer Vision*, pages 272–289. Springer, 2020.
- [40] P Majumder, M Mitra, and BB Chaudhuri. N-gram: a language independent approach to ir and nlp. In *International conference on universal knowledge and language*, 2002.
- [41] D. Martin, C. Fowlkes, D. Tal, and J. Malik. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In *Proceedings Eighth IEEE International Conference on Computer Vision. ICCV 2001*, volume 2, pages 416–423 vol.2, 2001. doi: 10.1109/ICCV.2001.937655.
- [42] Yusuke Matsui, Kota Ito, Yuji Aramaki, Azuma Fujimoto, Toru Ogawa, T. Yamasaki, and Kiyoharu Aizawa. Sketch-based manga retrieval using manga109 dataset. *Multimedia Tools and Applications*, 76:21811 – 21838, 2015. URL <https://api.semanticscholar.org/CorpusID:8887614>.
- [43] X. Pan, T. Ye, Z. Xia, S. Song, and G. Huang. Slide-transformer: Hierarchical vision transformer with local self-attention. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2082–2091, Los Alamitos, CA, USA, jun 2023. IEEE Computer Society. doi: 10.1109/CVPR52729.2023.00207. URL <https://doi.ieee.org/10.1109/CVPR52729.2023.00207>.
- [44] Wenzhe Shi, Jose Caballero, Ferenc Huszar, 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 (CVPR)*, 2016.
- [45] Robin Strudel, Ricardo Garcia, Ivan Laptev, and Cordelia Schmid. Segmenter: Transformer for semantic segmentation. *arXiv preprint arXiv:2105.05633*, 2021.
- [46] Long Sun, Jinshan Pan, and Jinhui Tang. ShuffleMixer: An efficient convnet for image super-resolution. In *NeurIPS*, 2022.
- [47] Long Sun, Jiangxin Dong, Jinhui Tang, and Jinshan Pan. Spatially-adaptive feature modulation for efficient image super-resolution. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 13190–13199, October 2023.
- [48] Ying Tai, Jian Yang, and Xiaoming Liu. Image super-resolution via deep recursive residual network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.
- [49] Mingxing Tan and Quoc Le. EfficientNetV2: Smaller models and faster training. In *In Proceedings of the International Conference on Machine Learning (ICML)*, 2021.
- [50] Radu Timofte, Eirikur Agustsson, Luc Van Gool, Ming-Hsuan Yang, Lei Zhang, Bee Lim, Sanghyun Son, Heewon Kim, Seungjun Nah, Kyoung Mu Lee, et al. Ntire 2017 challenge on single image super-resolution: Methods and results. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, pages 1110–1121, 2017. doi: 10.1109/CVPRW.2017.149.
- [51] Tong Tong, Gen Li, Xiejie Liu, and Qinquan Gao. Image super-resolution using dense skip connections. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2017.

- [52] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fdb053c1c4a845aa-Paper.pdf.
- [53] Jiamian Wang, Huan Wang, Yulun Zhang, Yun Fu, and Zhiqiang Tao. Iterative soft shrinkage learning for efficient image super-resolution. *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 12556–12565, 2023. URL <https://api.semanticscholar.org/CorpusID:257622691>.
- [54] Longguang Wang, Xiaoyu Dong, Yingqian Wang, Xinyi Ying, Zaiping Lin, Wei An, and Yulan Guo. Exploring sparsity in image super-resolution for efficient inference. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.
- [55] Zhou Wang, A.C. Bovik, H.R. Sheikh, and E.P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):600–612, 2004. doi: 10.1109/TIP.2003.819861.
- [56] Enze Xie, Wenhui Wang, Zhiding Yu, Anima Anandkumar, Jose M Alvarez, and Ping Luo. Segformer: Simple and efficient design for semantic segmentation with transformers. In *Neural Information Processing Systems (NeurIPS)*, 2021.
- [57] Bowen Yin, Xuying Zhang, Qibin Hou, Bo-Yuan Sun, Deng-Ping Fan, and Luc Van Gool. Camoformer: Masked separable attention for camouflaged object detection. *arXiv preprint arXiv:2212.06570*, 2022.
- [58] 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 *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.
- [59] Roman Zeyde, Michael Elad, and Matan Protter. On single image scale-up using sparse-representations. In *Curves and Surfaces*, pages 711–730, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg. ISBN 978-3-642-27413-8.
- [60] Kai Zhang, Yawei Li, Wangmeng Zuo, Lei Zhang, Luc Van Gool, and Radu Timofte. Plug-and-play image restoration with deep denoiser prior. *IEEE TPAMI*, 2021.
- [61] Qiming Zhang, Jing Zhang, Yufei Xu, and Dacheng Tao. Vision transformer with quadrangle attention. *arXiv preprint arXiv:2303.15105*, 2023.
- [62] Xiaoming Zhang, Tianrui Li, and Xiaole Zhao. Boosting single image super-resolution via partial channel shifting. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 13223–13232, 2023.
- [63] Xindong Zhang, Hui Zeng, and Lei Zhang. Edge-oriented convolution block for real-time super resolution on mobile devices. In *ACM MM*, 2021.
- [64] Xindong Zhang, Hui Zeng, Shi Guo, and Lei Zhang. Efficient long-range attention network for image super-resolution. *arXiv preprint arXiv:2203.06697*, 2022.
- [65] Yulun Zhang, Kunpeng Li, Kai Li, Lichen Wang, Bineng Zhong, and Yun Fu. Image super-resolution using very deep residual channel attention networks. In *ECCV*, 2018.
- [66] Yulun Zhang, Yapeng Tian, Yu Kong, Bineng Zhong, and Yun Fu. Residual dense network for image super-resolution. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.
- [67] Yulun Zhang, Huan Wang, Can Qin, and Yun Fu. Learning efficient image super-resolution networks via structure-regularized pruning. In *International Conference on Learning Representations*, 2021.
- [68] Hengyuan Zhao, Xiangtao Kong, Jingwen He, Yu Qiao, and Chao Dong. Efficient image super-resolution using pixel attention. In *Proceedings of the IEEE Conference on European Conference on Computer Vision (ECCV) Workshops*, 2020.
- [69] Yupeng Zhou, Zhen Li, Chun-Le Guo, Song Bai, Ming-Ming Cheng, and Qibin Hou. Srformer: Permuted self-attention for single image super-resolution. *arXiv preprint arXiv:2303.09735*, 2023.

A Impact of number of Blocks, Channels, Heads and Depths

In this section, we analyze how the performance of our proposed model changes based on the number of blocks, heads, channels and depths.

Table A: Performance difference of LMLT based on the number of blocks.

#Block	#Params	#FLOPs	#GPU Mem	#AVG Time	Set5	Set14	B100	Urban100	Manga109
4	122K	30G	323.52M	29.75ms	37.88/0.9601	33.40/0.9166	32.06/0.8984	31.49/0.9217	38.47/0.9765
6	181K	44G	323.77M	43.51ms	37.94/0.9601	33.52/0.9175	32.15/0.8995	31.82/0.9253	38.74/0.9771
8	239K	59G	324.01M	57.37ms	38.01/0.9606	33.59/0.9183	32.19/0.8999	32.04/0.9273	38.90/0.9775
10	298K	73G	324.26M	70.55ms	38.05/0.9608	33.66/0.9188	32.22/0.9003	32.17/0.9286	39.00/0.9778
12	357K	88G	324.5M	84.22ms	38.05/0.9608	33.65/0.9187	32.24/0.9006	32.31/0.9298	39.10/0.9780

Impact of Number of Blocks. First, We evaluate the performance by varying the number of blocks to 4, 6, 8, 10, and 12. Experiments are conducted on $\times 2$ scale and the performance is evaluated using benchmark datasets, and analyzed in terms of the number of parameters, FLOPs, GPU memory usage, and average inference time.

As shown in Table A, the increase in the number of parameters, FLOPs and inference time tends to be proportional to the number of blocks, and performance also gradually improves. For the Manga109 [42] dataset, as the number of blocks increases from 4 to 12 in increments of 2, PSNR increases by 0.27 db, 0.16 db, 0.10 db, and 0.10 db, respectively. Interestingly, despite the increase in the number of blocks from 4 to 12, the GPU memory usage remains almost unchanged. While the number of parameters nearly triples, the GPU memory usage remains stable, 323.5M to 324.5M. We observe the overall increase in PSNR with the increase in the number of blocks and designate the model with 8 blocks as LMLT-Tiny and the model with 12 blocks as LMLT-Small.

Table B: Performance difference of LMLT based on the number of channels.

#Channel	#Params	#FLOPs	#GPU MEM	#AVG Time	Set5	Set14	B100	Urban100	Manga109
24	109K	27G	255.77M	38.34ms	37.91/0.9602	33.44/0.9169	32.09/0.8988	31.62/0.9231	38.58/0.9768
36	239K	59G	324.01M	57.37ms	38.01/0.9606	33.59/0.9183	32.19/0.8999	32.04/0.9273	38.90/0.9775
48	420K	103G	460.32M	65.66ms	38.06/0.9609	33.67/0.9189	32.25/0.9007	32.33/0.9299	39.14/0.9780
60	652K	158G	567.75M	81.64ms	38.10/0.9610	33.76/0.9201	32.28/0.9012	32.52/0.9316	39.24/0.9783
72	935K	226G	684.82M	108.74ms	38.17/0.9612	33.83/0.9205	32.32/0.9016	32.65/0.9329	39.36/0.9786
84	1,270K	306G	717.31M	123.07ms	38.18/0.9612	33.96/0.9212	32.33/0.9017	32.75/0.9336	39.41/0.9786

Impact of Number of Channels. Next, we evaluate how performance changes with the number of channels. Similar to the performance evaluation based on the number of blocks, this experiment evaluates performance using benchmark datasets, the number of parameters, FLOPs, GPU memory usage, and average inference time as performance metrics.

As shown in Table B, LMLT’s performance increases with channels, along with parameters and FLOPs. However, unlike the variations in the number of blocks, increasing the number of channels results in a more significant increase in the number of parameters, FLOPs, and memory usage. Inference time, however, increases proportionally with the number of channels. For instance, with 36 channels, the average inference time is 57.16ms, and when doubled, it requires approximately 108.74ms, nearly twice the time. As the number of channels increases from 24 to 84 in increments of 12, the PSNR on the Urban100 [23] dataset increases by 0.42 db, 0.29 db, 0.19 db, 0.13 db, and 0.10 db, respectively. Based on the overall performance increase, we designate the model with 60 channels as the Base model and the model with 84 channels as the Large model. In this context, the Small model has an inference time about 3ms longer than the Base model, but it has fewer parameters, lower memory usage, and fewer FLOPs, thus justifying its designation.

Impact of Number of Heads. In this paragraph, we compare the performance differences based on the number of heads. In our model, as the number of heads decreases, the channel and the number of downsizing operations for each head decrease. For example, in our baseline with 4 heads and 36 channels, the lowest head has a total of 9 channels and is pooled 3 times. However, if there are 2 heads, the lowest head has 18 channels and is pooled once. Additionally, the maximum pooling times and the number of heads are related to the number of windows and the amount of self-attention computation. According to equation 1, as the number of heads decreases, the computation increases. As a result, as the number of heads decreases, the number of parameters, FLOPs, and GPU memory usage increase.

As shown in Table C, the performance with 4 heads and 3 heads is similar across all scales and test datasets. However, when the number of heads is reduced to 1, the performance drops significantly. This difference is particularly noticeable in the Urban100 [23] dataset, where at scale $\times 2$, the performance with 4 heads is 32.04 db, whereas with 1 head, it drops to 31.93 db, a decrease of 0.11 db. Additionally, when the scale is $\times 3$ and $\times 4$, the PSNR decreases by 0.05 dB and 0.04 dB, respectively. This indicates that even if the spatial size of all

Table C: Performance difference of LMLT based on the number of heads. Chan is the number of channels in each heads. Best results are highlighted.

Scale	#Heads	#Chan	#Params	#FLOPs	#Acts	#GPU	Set5	Set14	B100	Urban100	Manga109
$\times 2$	1	36	270K	75G	845M	437.21M	38.00/0.9606	33.58/0.9179	32.17/0.8997	31.93/0.9260	38.83/0.9774
	2	18	250K	64G	717M	385.96M	38.01 /0.9606	33.59 /0.9180	32.18/ 0.8999	32.02/0.9270	38.87/ 0.9776
	3	12	243K	60G	646M	346.71M	38.00/0.9606	33.59 /0.9182	32.19 / 0.8999	32.02/ 0.9273	38.88/0.9775
	4	9	239K	59G	603M	324.01M	38.01 /0.9606	33.59 /0.9183	32.19 / 0.8999	32.04 / 0.9273	38.90 /0.9775
$\times 3$	1	36	275K	35G	396M	205.52M	34.37/0.9271	30.39 / 0.8431	29.11/0.8054	28.05/0.8495	33.71/0.9449
	2	18	255K	30G	336M	181.14M	34.37/0.9271	30.37/0.8427	29.11/0.8056	28.05/0.8496	33.73 / 0.9450
	3	12	248K	29G	303M	161.90M	34.41 / 0.9273	30.37/0.8426	29.12 / 0.8059	28.09/0.8502	33.73/0.9449
	4	9	244K	28G	283M	151.96M	34.36/0.9271	30.37/0.8427	29.12 /0.8057	28.10 / 0.8503	33.72/0.9448
$\times 4$	1	36	282K	19G	213M	111.28M	32.14/0.8943	28.65 / 0.7826	27.60/0.7366	26.04/0.7825	30.57/0.9080
	2	18	261K	17G	181M	98.69M	32.18/ 0.8948	28.63/ 0.7826	27.60/ 0.7370	26.07/0.7839	30.59/0.9085
	3	12	254K	16G	163M	87.04M	32.19 /0.8947	28.63/0.7821	27.60/0.7367	26.08 /0.7834	30.58/0.9080
	4	9	251K	15G	152M	81.44M	32.19 /0.8947	28.64/0.7823	27.60/0.7369	26.08 / 0.7838	30.60 / 0.9083

Table D: Performance difference of LMLT based on the number of depths. Best results are highlighted.

Scale	#Depths	#Params	#FLOPs	#Acts	#AVG Time	Set5	Set14	B100	Urban100	Manga109
$\times 2$	1	239K	59G	603M	57.37ms	38.01/0.9606	33.59/0.9183	32.19/0.8999	32.04/0.9273	38.90 /0.9775
	2	254K	63G	911M	70.93ms	38.01/0.9606	33.61 / 0.9185	32.20 / 0.9000	32.06/0.9274	38.89/ 0.9776
	3	268K	67G	1219M	84.50ms	38.01/ 0.9607	33.59/0.9180	32.19/ 0.9000	32.08 / 0.9276	38.89/ 0.9776
$\times 3$	1	244K	28G	283M	31.06ms	34.36/0.9271	30.37/0.8427	29.12 / 0.8057	28.10/0.8503	33.72/0.9448
	2	259K	30G	427M	38.20ms	34.39/ 0.9275	30.38/ 0.8429	29.11/0.8056	28.11 / 0.8507	33.75 / 0.9451
	3	273K	32G	570M	48.75ms	34.40 /0.9274	30.39 /0.8428	29.11/0.8056	28.08/0.8501	33.73/0.9449
$\times 4$	1	251K	15G	152M	23.54ms	32.19/0.8947	28.64/0.7823	27.60/0.7369	26.08/0.7838	30.60/0.9083
	2	265K	16G	230M	33.27ms	32.20/0.8949	28.65/0.7823	27.60/0.7369	26.08/0.7839	30.58/0.9083
	3	279K	17G	307M	44.07ms	32.23 / 0.8954	28.66 / 0.7825	27.61 /0.7369	26.10 / 0.7845	30.59/ 0.9084

features is maintained with a single head, even though the number of parameters, FLOPs, and channels per head increase, the inability to capture information from other windows can lead to a decline in performance.

Impact of Number of Depths. Additionally, we examine how the performance changes when we add more attention modules to our model. The proposed LMLT connects self-attention layers in parallel, where self-attention is calculated in lower heads and then connected to the upper layers. However, a different approach, like other models [32, 69], could be to calculate self-attention multiple times(i.e., in series) before sending it to the upper heads, thus mixing serial and parallel connections.

However, our experimental results indicate that calculating self-attention multiple times in a single head before sending it to the upper heads is not an effective choice. As shown in Table D, increasing the self-attention calculations from once to three times increases the inference time by 87.7% on the Urban100 [23] dataset at $\times 4$ scale, but the PSNR only improves by 0.02 dB. Similar trends are observed in other datasets and scales, showing minimal differences in PSNR and SSIM performance. This demonstrates that having one head composed of serial self-attention layers and connecting heads in parallel does not yield good performance relative to inference time.

B Effects of Low-to-high connection and Pooling

Difference between with and without Low-to-high connection In the Table 4, we confirm performance differences when the low-to-high connection is not applied to LMLT. Inspired by this, we also apply low-to-high connections between heads in SAFMN [47] and verify the experimental results. Table E shows that adding low-to-high connection to the upper head in SAFMN [47] does not yield significant performance differences. Moreover, at the $\times 4$ scale, the SSIM for the Urban100 [23] and Set5 [3] datasets decreases by 0.0010 and 0.0011, respectively, indicating a reduction in performance.

We then visualize the features of LMLT-Tiny to understand the effect of the low-to-high connection. Each column of Figure A illustrates the original image, the aggregated feature visualization of LMLT-Tiny combining all heads A(a), and the aggregated feature visualization of LMLT-Tiny without low-to-high connection A(b). In A(b), the images show pronounced boundaries in areas such as stairs, buildings, and the sky. In contrast, A(a) shows these boundaries as less pronounced. This demonstrates that the low-to-high connection can address the border communication issues inherent in WSA.

Difference between with and without Pooling We analyze the impact of pooling on the performance of super-resolution. As observed in Table 5, even though pooling preserves spatial information, the overall performance decreases when it is not applied. We investigate the reason behind this through feature visualization. Figure B visualizes the features when no pooling is applied to any head in LMLT. The leftmost image is the

Table E: The comparison table between SAFMN and its variant with low-to-high element-wise sum added. For each model, the better results are highlighted in **bold**.

Scale	Method	Set5	Set14	B100	Urban100	Manga109
$\times 2$	SAFMN [47]	38.00/0.9605	33.54/0.9177	32.16/0.8995	31.84/0.9256	38.71/0.9771
	SAFMN [47] w/ sum	37.99/0.9604	33.52/0.9174	32.17/0.8996	31.86/0.9257	38.77/0.9773
$\times 3$	SAFMN [47]	34.34/0.9267	30.33/0.8418	29.08/0.8048	27.95/0.8474	33.52/0.9437
	SAFMN [47] w/ sum	34.34/0.9269	30.32/0.8418	29.09/0.8049	27.96/0.8476	33.55/0.9438
$\times 4$	SAFMN [47]	32.18/0.8948	28.60/0.7813	27.58/0.7359	25.97/0.7809	30.43/0.9063
	SAFMN [47] w/ sum	32.10/0.8937	28.59/0.7812	27.58/0.7358	25.96/0.7799	30.44/0.9063

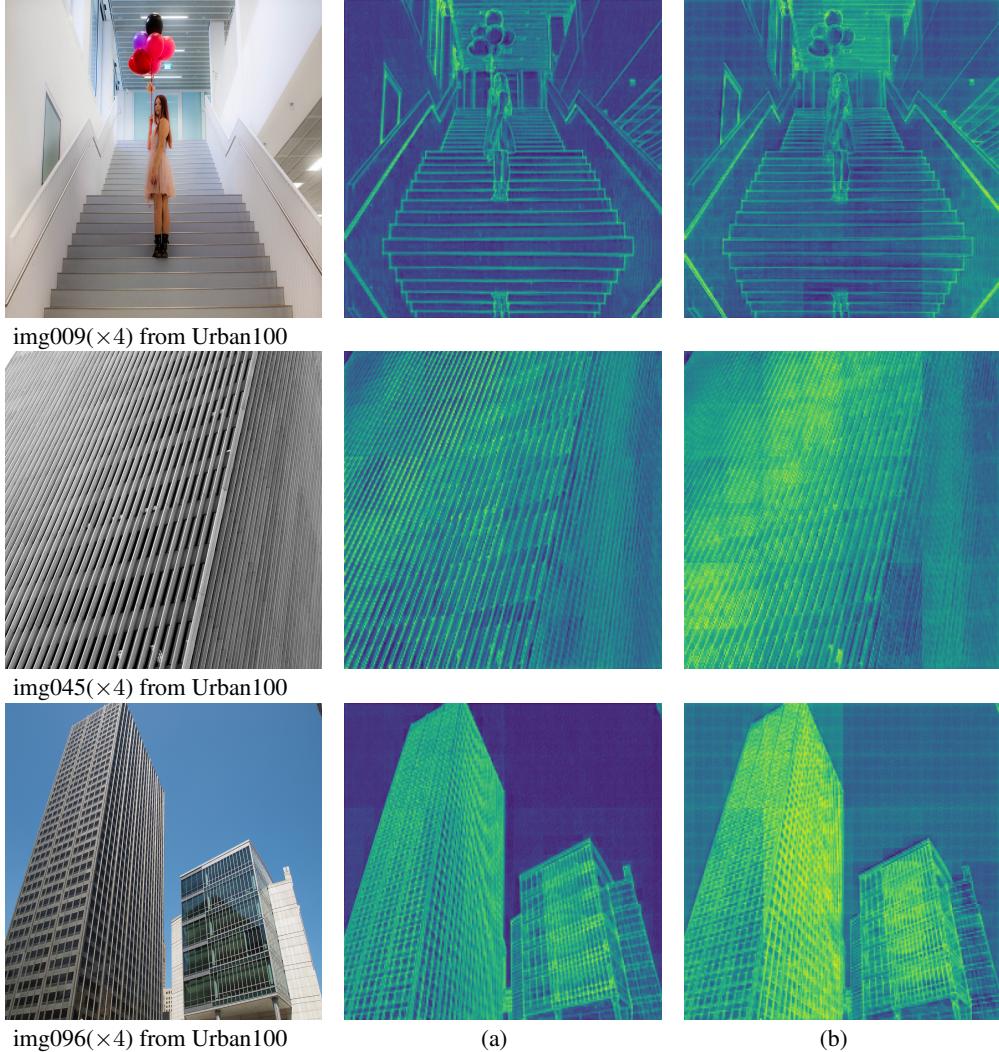


Figure A: Visualization of features with low-to-high connection(a) and without connection(b) on Urban100 $\times 4$. As shown in the images, without the low-to-high connection, the boundaries between windows are clearly visible.

original Urban100 [23] image. Figure B(a) shows the aggregated features of all heads in LMLT-Tiny. Column Figure B(b) visualizes the features without pooling, and Figure B(c) visualizes the features without both pooling and merging, all at the $\times 4$ scale. In B(b) and B(c), grid patterns are evident across the images, indicating that the disadvantages of being limited to local windows outweigh the benefits of maintaining the original spatial size.

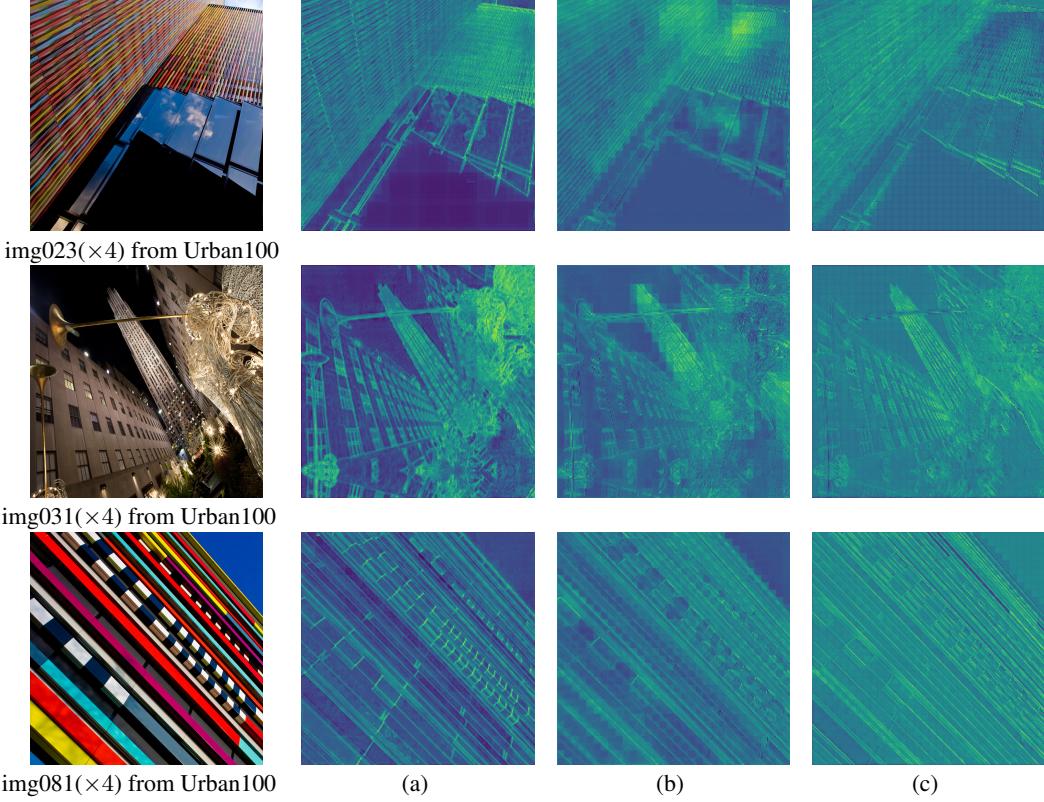


Figure B: Comparison without pooling on Urban100 $\times 4$ scale. From left to right: (a) LMLT with pooling applied, (b) without any pooling, (c) without pooling and without multiplication by activation. In (b) and (c), the boundaries between windows are visible.

Table F: Performance difference of LMLT with GELU and without GELU. The better results are highlighted in **bold**.

Scale	Ablation	#Channel	Set5	Set14	B100	Urban100	Manga109
$\times 2$	LMLT	36	38.01/0.9606	33.59/0.9183	32.19/0.8999	32.04/0.9273	38.90 /0.9775
	LMLT <i>w/o</i> GELU	36	38.03/0.9606	33.60/0.9184	32.19/ 0.9000	32.05/0.9272	38.91/0.9776
	LMLT	60	38.10/0.9610	33.76/0.9201	32.28/ 0.9012	32.52/0.9316	39.24/0.9783
	LMLT <i>w/o</i> GELU	60	38.10/0.9610	33.80/0.9200	32.28/0.9011	32.51/0.9315	39.26/0.9783
$\times 3$	LMLT	36	34.36/0.9271	30.37/0.8427	29.12/0.8057	28.10/0.8503	33.72/0.9448
	LMLT <i>w/o</i> GELU	36	34.37/0.9272	30.36/0.8425	29.11/0.8057	28.08/0.8502	33.71/0.9447
	LMLT	60	34.58/0.9285	30.53/0.8458	29.21/0.8084	28.48/0.8581	34.18/0.9477
	LMLT <i>w/o</i> GELU	60	34.53/0.9283	30.51/0.8457	29.20/0.8080	28.45/0.8576	34.17/0.9476
$\times 4$	LMLT	36	32.19/0.8947	28.64/0.7823	27.60/0.7369	26.08/0.7838	30.60/0.9083
	LMLT <i>w/o</i> GELU	36	32.23/0.8949	28.62/0.7820	27.60/0.7369	26.08/0.7836	30.59/0.9082
	LMLT	60	32.38/0.8971	28.79/0.7859	27.70/0.7403	26.44/0.7947	31.09/0.9139
	LMLT <i>w/o</i> GELU	60	32.39/0.8973	28.78/0.7858	27.69/0.7399	26.39/0.7934	31.04/0.9132

C Impact of Activation function

In Table 6, we discuss that not applying the activation function GeLU [20] might improve performance. Therefore, we experiment with LMLT and LMLT without GeLU [20] across various scales and channels to confirm the results.

Table F shows the results for our LMLT and the model without the activation function across different scales and channels. As shown, with 36 channels, there is minimal performance difference across all scales, with the largest being a 0.04 higher PSNR on the Set5 [3] $\times 4$ scale when GeLU [20] is removed. However, when expanded to 60 channels, our LMLT performs better on most benchmark datasets for both $\times 3$ and $\times 4$ scales. Specifically, on

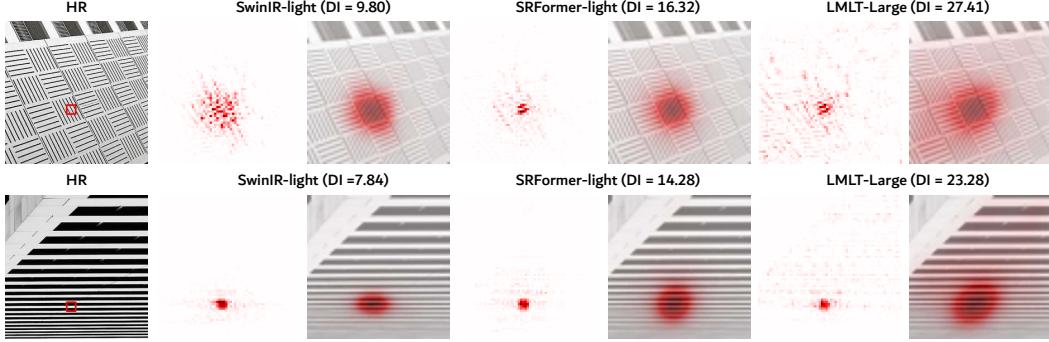


Figure C: The LAM results of SwinIR-Light [32], SRFormer-Light [69], and our proposed model(LMLT-Large). As shown in the figure, our proposed model references a broader range of pixels when reconstructing the image.

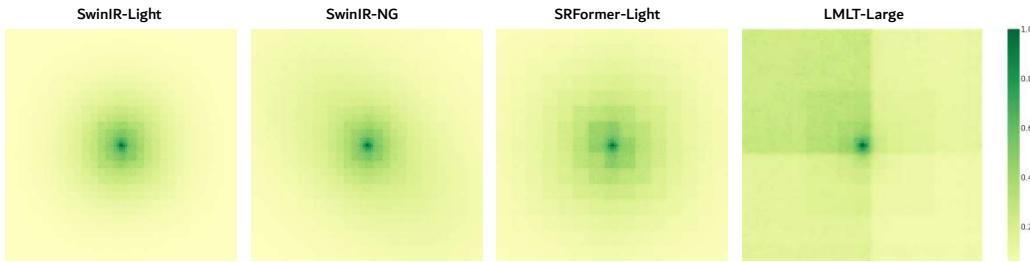


Figure D: The ERF visualizations of SwinIR-Light [32], SwinIR-NG [8], SRFormer-Light [69], and the proposed model (LMLT-Large). The darker areas are more widely distributed, indicating a larger ERF, and the figure shows that the proposed model effectively utilizes global information.

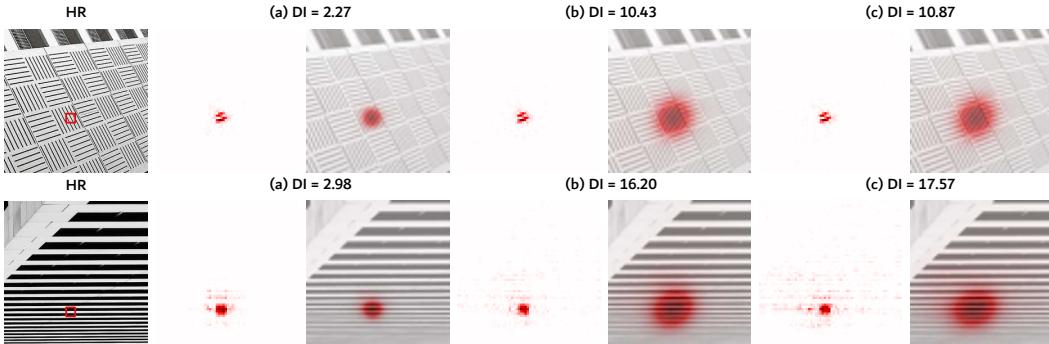


Figure E: Visualization of LAM-Tiny. From left to right: (a) LMLT-Tiny without pooling, (b) LMLT-Tiny without low-to-high connection and (c) LMLT-Tiny.

the $\times 4$ scale of the Urban100 [23] dataset, PSNR and SSIM are higher by 0.05 dB and 0.0013, respectively. This demonstrates that adding GeLU [20] after aggregating features is more beneficial for performance improvement.

D LAM and ERF Comparisons

To verify whether the proposed LMLT exhibits a wider receptive field, we utilize local attribution map (LAM) [16] and effective receptive field (ERF) [38]. Specifically, we use LAM to show that our proposed LMLT-Large has a wider receptive field compared to SwinIR-Light [32] and SRFormer-Light [69]. Detailed visualizations are presented in Figure C. Additionally, SwinIR-NG [8] is included for comparison, and we visualize the ERF. The detailed results are shown in Figure D. Through these two analyses, we demonstrate that our proposed model exhibits a wider receptive field than existing ViT-based SR models.

Table G: Comparisons with existing methods. Best and second-best performance are in **red** and **blue**, and third-best is underlined. Unreported results are left blank.

Scale	Method	#Params	#FLOPs	#Acts	Set5	Set14	B100	Urban100	Manga109
$\times 2$	CARN-M [1]	412K	91G	655M	37.53/0.9583	33.26/0.9141	31.92/0.8960	31.23/0.9193	-
	CARN [1]	1,592K	223G	<u>522M</u>	37.76/0.9590	33.52/0.9166	32.09/0.8978	31.92/0.9256	-
	EDSR-baseline [33]	1,370K	316G	563M	37.99/0.9604	33.57/0.9175	32.16/0.8994	31.98/0.9272	38.54/0.9769
	PAN [68]	261K	71G	677M	<u>38.00</u> /0.9605	33.59/0.9181	<u>32.18</u> /0.8997	32.01/0.9273	38.70/0.9773
	LAPAR-A [30]	548K	171G	656M	38.01 /0.9605	33.62/0.9183	32.19 /0.8999	32.10/0.9283	38.67/0.9772
	ECBSR-M16C64 [63]	596K	137G	252M	37.90/ 0.9615	33.34/0.9178	32.10/ 0.9018	31.71/0.9250	-
	SMSR [54]	985K	132G	-	<u>38.00</u> /0.9601	<u>33.64</u> /0.9179	32.17/0.8990	32.19 /0.9284	38.76/0.9771
	ShuffleMixer [46]	394K	91G	832M	38.01 /0.9606	<u>33.63</u> /0.9180	32.17/0.8995	31.89/0.9257	38.83/0.9774
	SAFMN [47]	228K	52G	299M	38.00/0.9605	33.54/0.9177	32.16/0.8995	31.84/0.9256	38.71/0.9771
	LMLT-Tiny(Ours)	239K	59G	603M	38.01 /0.9606	33.59/0.9183	32.19 /0.8999	32.04/0.9273	38.90 /0.9775
	LMLT-Small(Ours)	357K	88G	898M	38.05 /0.9608	<u>33.65</u> /0.9187	<u>32.24</u> /0.9006	<u>32.31</u> /0.9298	<u>39.10</u> /0.9780
$\times 3$	CARN-M [1]	415K	46G	327M	33.99/0.9236	30.08/0.8367	28.91/0.8000	27.55/0.8385	-
	CARN [1]	1,592K	119G	<u>268M</u>	34.29/0.9255	30.29/0.8407	29.06/0.8034	28.06/0.8493	-
	EDSR-baseline [33]	1,555K	160G	285M	<u>34.37</u> /0.9270	30.28/0.8417	29.09/0.8052	28.15/0.8527	33.45/0.9439
	PAN [68]	261K	39G	340M	34.40 /0.9271	30.36/0.8423	29.11/0.8050	28.11/0.8511	33.61 /0.9448
	LAPAR-A [30]	594K	114G	505M	34.36/0.9267	<u>30.34</u> /0.8421	<u>29.11</u> /0.8054	28.15/0.8523	33.51/0.9441
	SMSR [54]	993K	68G	-	<u>34.40</u> /0.9270	30.33/0.8412	29.10/0.8050	28.25 /0.8536	33.68/0.9445
	ShuffleMixer [46]	415K	43G	404M	34.40 /0.9272	<u>30.37</u> /0.8423	<u>29.12</u> /0.8051	28.08/0.8498	<u>33.69</u> /0.9448
	SAFMN [47]	233K	23G	134M	34.34/0.9267	30.33/0.8418	29.08/0.8048	27.95/0.8474	33.52/0.9437
	LMLT-Tiny(Ours)	244K	28G	283M	34.36/0.9271	<u>30.37</u> /0.8427	<u>29.12</u> /0.8057	28.10/0.8503	33.72 /0.9448
	LMLT-Small(Ours)	361K	41G	421M	<u>34.50</u> /0.9280	<u>30.47</u> /0.8446	<u>29.16</u> /0.8070	<u>28.29</u> /0.8544	33.99/0.9464
$\times 4$	CARN-M [1]	412K	33G	227M	31.92/0.8903	28.42/0.7762	27.44/0.7304	25.62/0.7694	-
	CARN [1]	1,592K	91G	194M	32.13/0.8937	28.60/0.7806	27.58/0.7349	26.07/0.7837	-
	EDSR-baseline [33]	1,518K	114G	202M	32.09/0.8938	28.58/0.7813	27.57/0.7357	26.04/0.7849	30.35/0.9067
	PAN [68]	272K	<u>28G</u>	238M	<u>32.13</u> /0.8948	28.61/0.7822	27.59/0.7363	26.11/0.7854	30.51/0.9095
	LAPAR-A [30]	659K	94G	452M	32.15/0.8944	28.61/0.7818	27.61 /0.7366	26.14 /0.7871	30.42/0.9074
	ECBSR-M16C64 [63]	603K	35G	64M	31.92/0.8946	28.34/0.7817	27.48/0.7393	25.81/0.7773	-
	SMSR [54]	1006K	42G	-	32.12/0.8932	28.55/0.7808	27.55/0.7351	<u>26.11</u> /0.7868	30.54/0.9085
	ShuffleMixer [46]	411K	28G	269M	<u>32.21</u> /0.8953	28.66 /0.7827	27.61 /0.7366	26.08/0.7835	30.65 /0.9093
	SAFMN [47]	240K	14G	77M	32.18/0.8948	28.60/0.7813	27.58/0.7359	25.97/0.7809	30.43/0.9063
	LMLT-Tiny(Ours)	251K	15G	152M	<u>32.19</u> /0.8947	<u>28.64</u> /0.7823	<u>27.60</u> /0.7369	26.08/0.7838	<u>30.60</u> /0.9083
	LMLT-Small(Ours)	368K	23G	227M	<u>32.31</u> /0.8968	<u>28.74</u> /0.7846	<u>27.66</u> /0.7387	<u>26.26</u> /0.7894	<u>30.87</u> /0.9117

Additionally, we compare the LAM of our proposed model, LMLT-Tiny (Figure E(c)), with a version of the model that does not include pooling for each head (Figure E(a)), and a version where the low-to-high connection is removed (Figure E(b)), demonstrating that our proposed model effectively references a broader region. The results show that, even when the spatial size of the model is maintained without pooling, it fails to process information from a wider area. Moreover, the low-to-high connection proves to be effective in enabling the model to capture information from a larger region.

E CCM : Convolutional Channel Mixer

CCM instead of MLP. Since the feed-forward network (FFN) in the original transformer [52] is a fully connected layer, we assume that using it in ViT might disrupt the spatial information of the features. Therefore, we apply the convolutional channel mixer (CCM) [47] instead, an FFN based on FMBConv [49], to preserve spatial information. CCM is a module that mixes each convolution channel. Specifically, the features pass through two convolution layers. The first layer has a 3×3 kernel and expands the channels. Then, GELU [20] is applied for non-linear mapping. Finally, a convolution layer with a 1×1 kernel restores the channels to their original state. In our method, the features pass through Layer Normalization [2], LMLT, and another Layer Normalization before being input to CCM [47]. Detailed structure can be seen in Figure F.

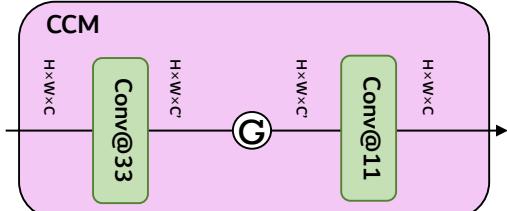


Figure F: CCM(Convolutional Channel Mixer) proposed in SAFMN [47].

F Comparisons on LMLT with Other Methods

Image Reconstruction comparisons Here, We first compare the LMLT-Tiny and LMLT-Small with CARN-m, CARN [1], EDSR-baseline [33], PAN [68], LAPAR-A [30], ECBSR-M16C64 [63], SMSR [54], Shuffle-

Table H: The memory consumption and inference times are reported. All experiments were conducted on a single RTX 3090 GPU.

Scale	Method	#GPU Mem [M]	#Avg Time [ms]	Set5	Set14	B100	Urban100	Manga109
$\times 2$	CARN-M [1]	2707.82	67.56	37.53/0.9583	33.26/0.9141	31.92/0.8960	31.23/0.9193	-
	CARN [1]	2716.80	73.55	37.76/0.9590	33.52/0.9166	32.09/0.8978	31.92/0.9256	-
	EDSR-baseline [33]	577.61	43.58	37.99/0.9604	33.57/0.9175	32.16/0.8994	31.98/0.9272	38.54/0.9769
	LAPAR-A [30]	1812.60	43.50	38.01/0.9605	33.62/0.9183	32.19/0.8999	32.10/0.9283	38.67/0.9772
	SAFMN [47]	259.56	33.61	38.00/0.9605	33.54/0.9177	32.16/0.8995	31.84/0.9256	38.71/0.9771
	LMLT-Tiny(Ours)	324.01	57.37	38.01/0.9606	33.59/0.9183	32.19/0.8999	32.04/0.9273	38.90/0.9775
	LMLT-Small(Ours)	324.5	84.22	38.05/0.9608	33.65/0.9187	32.24/0.9006	32.31/0.9298	39.10/0.9780
	IMDN [24]	795.96	31.87	38.00/0.9605	33.63/0.9177	32.19/0.8996	32.17/0.9283	38.88/0.9774
	HNCT [15]	1200.55	351.49	38.08/0.9608	33.65/0.9182	32.22/0.9001	32.22/0.9294	38.87/0.9774
	NGswin [8]	1440.40	375.19	38.05/0.9610	33.79/0.9199	32.27/0.9008	32.53/0.9324	38.97/0.9777
$\times 3$	LMLT-Base(Ours)	567.75	81.64	38.10/0.9610	33.76/0.9201	32.28/0.9012	32.52/0.9316	39.24/0.9783
	SwinIR-light [32]	1278.64	944.11	38.14/0.9611	33.86/0.9206	32.31/0.9012	32.76/0.9340	39.12/0.9783
	SRformer-Light [69]	1176.15	1006.48	38.23/0.9613	33.94/0.9209	32.36/0.9019	32.91/0.9353	39.28/0.9785
	LMLT-Large(Ours)	717.31	123.07	38.18/0.9612	33.96/0.9212	32.33/0.9017	32.75/0.9336	39.41/0.9786
	CARN-M [1]	1213.10	37.56	33.99/0.9236	30.08/0.8367	28.91/0.8000	27.55/0.8385	-
	CARN [1]	1222.08	41.08	34.29/0.9255	30.29/0.8407	29.06/0.8034	28.06/0.8493	-
	EDSR-baseline [33]	541.61	26.14	34.37/0.9270	30.28/0.8417	29.09/0.8052	28.15/0.8527	33.45/0.9439
	LAPAR-A [30]	1813.84	35.95	34.36/0.9267	30.34/0.8421	29.11/0.8054	28.15/0.8523	33.51/0.9441
	SAFMN [47]	114.70	17.38	34.34/0.9267	30.33/0.8418	29.08/0.8048	27.95/0.8474	33.52/0.9437
	LMLT-Tiny(Ours)	151.96	31.06	34.36/0.9271	30.37/0.8427	29.12/0.8057	28.10/0.8503	33.72/0.9448
	LMLT-Small(Ours)	152.5	44.22	34.50/0.9280	30.47/0.8446	29.16/0.8070	28.29/0.8544	33.99/0.9464
	IMDN [24]	364.68	14.01	34.36/0.9270	30.32/0.8417	29.09/0.8046	28.17/0.8519	33.61/0.9445
	HNCT [15]	545.64	117.20	34.47/0.9275	30.44/0.8439	29.15/0.8067	28.28/0.8557	33.81/0.9459
	NGswin [8]	696.97	168.49	34.52/0.9282	30.53/0.8456	29.19/0.8078	28.52/0.8603	33.89/0.9470
$\times 4$	LMLT-Base(Ours)	266.31	41.43	34.58/0.9285	30.53/0.8458	29.21/0.8084	28.48/0.8581	34.18/0.9477
	SwinIR-light [32]	587.63	287.96	34.62/0.9289	30.54/0.8463	29.20/0.8082	28.66/0.8624	33.98/0.9478
	SRformer-Light [69]	529.28	312.37	34.67/0.9296	30.57/0.8469	29.26/0.8099	28.81/0.8655	34.19/0.9489
	LMLT-Large(Ours)	338.36	58.68	34.64/0.9293	30.60/0.8471	29.26/0.8097	28.72/0.8626	34.43/0.9491
	CARN-M [1]	680.84	21.39	31.92/0.8903	28.42/0.7762	27.44/0.7304	25.62/0.7694	-
	CARN [1]	689.83	20.50	32.13/0.8937	28.60/0.7806	27.58/0.7349	26.07/0.7837	-
	EDSR-baseline [33]	492.39	19.86	32.09/0.8938	28.58/0.7813	27.57/0.7357	26.04/0.7849	30.35/0.9067
	LAPAR-A [30]	1811.47	32.24	32.15/0.8944	28.61/0.7818	27.61/0.7366	26.14/0.7871	30.42/0.9074
	SAFMN [47]	65.26	11.28	32.18/0.8948	28.60/0.7813	27.58/0.7359	25.97/0.7809	30.43/0.9063
	LMLT-Tiny(Ours)	81.44	23.54	32.19/0.8947	28.64/0.7823	27.60/0.7369	26.08/0.7838	30.60/0.9083
	LMLT-Small(Ours)	81.92	31.01	32.31/0.8968	28.74/0.7846	27.66/0.7387	26.26/0.7894	30.87/0.9117
	IMDN [24]	203.02	9.71	32.21/0.8948	28.58/0.7811	27.56/0.7353	26.04/0.7838	30.45/0.9075
	HNCT [15]	312.72	69.61	32.31/0.8957	28.71/0.7834	27.63/0.7381	26.20/0.7896	30.70/0.9112
	NGswin [8]	372.94	118.13	32.33/0.8963	28.78/0.7859	27.66/0.7396	26.45/0.7963	30.80/0.9128
$\times 4$	LMLT-Base(Ours)	144.00	26.15	32.38/0.8971	28.79/0.7859	27.70/0.7403	26.44/0.7947	31.09/0.9139
	SwinIR-light [32]	342.46	176.76	32.44/0.8976	28.77/0.7858	27.69/0.7406	26.47/0.7980	30.92/0.9151
	SRformer-Light [69]	320.95	180.42	32.51/0.8988	28.82/0.7872	27.73/0.7422	26.67/0.8032	31.17/0.9165
	LMLT-Large(Ours)	185.68	34.07	32.48/0.8987	28.87/0.7879	27.75/0.7421	26.63/0.8001	31.32/0.9163

Mixer [46], and SAFMN [47]. Table G shows that our LMLT significantly reduces number of parameters and computation overheads while achieving substantial performance gains on various datasets. LMLT-Small performs well on most datasets, and the LMLT-Tiny also performs second and third best on the BSD100 [41] and Manga109 [42] datasets, except for the Manga109 $\times 4$ SSIM [55]. In particular, the number of parameters and FLOPs are the second smallest after SAFMN [47].

Memory and Running time Comparisons In this paragraph, we present the memory usage and average inference time of our proposed LMLT compared to other super-resolution methods. Similar to the experimental setup in Table 2, #GPU Mem represents the maximum memory usage during inference, measured using PyTorch’s `torch.cuda.max_memory_allocated()`. #AVG Time indicates the average time taken to upscale a total of 50 random images by $\times 2$, $\times 3$, and $\times 4$ scales. The experiments were conducted three times, and the average inference time is reported. Each random image has sizes of 640×360 for $\times 2$ scale, 427×240 for $\times 3$ scale, and 320×180 for $\times 4$ scale.

As shown in Table H, our proposed LMLT-Tiny uses less memory at all scales compared to all models except SAFMN [47]. Although LMLT-Small requires more inference time compared to other models, its GPU usage is almost similar to LMLT-Tiny, and its performance is significantly superior as demonstrated in Table G.

Qualitative Comparisons In this paragraph, we examine the qualitative comparisons of the LMLT-Tiny model and other models on the Urban100 [23] $\times 4$ scale. The comparison includes CARN [1], EDSR [33], PAN [68], ShuffleMixer [46], and SAFMN [47]. The results can be seen in Figure G. As mentioned in section 4.1, we observe that our model reconstructs images with continuous stripes better than other models.

Additionally, we compare our proposed models LMLT-Base and LMLT-Large with IMDN [24], NGswin [8], SwinIR-light [32], and SwinIR-NG [8] on the Manga109 [42] dataset at $\times 4$ scale. As explained earlier in section 4.1, our model shows strength in areas with continuous lines compared to other models. Figure H illustrates the differences between our LMLT-Base, LMLT-Large and other state-of-the-arts models.

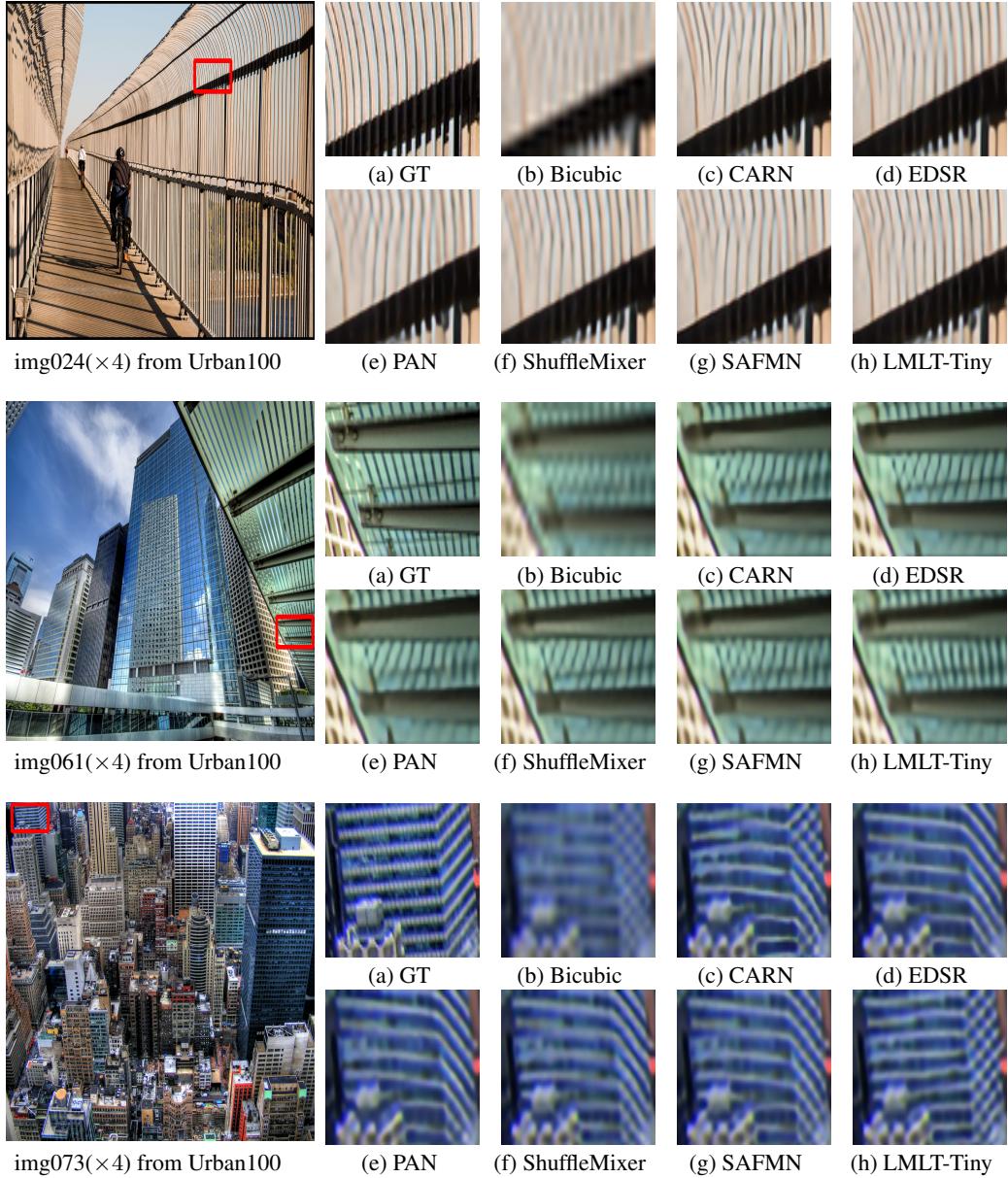


Figure G: Visual comparisons for $\times 4$ SR on Urban100 dataset. Compared with the results in (c) to (g), the Ours(LMLT-Tiny, (h)) restore much more accurate and clear images.

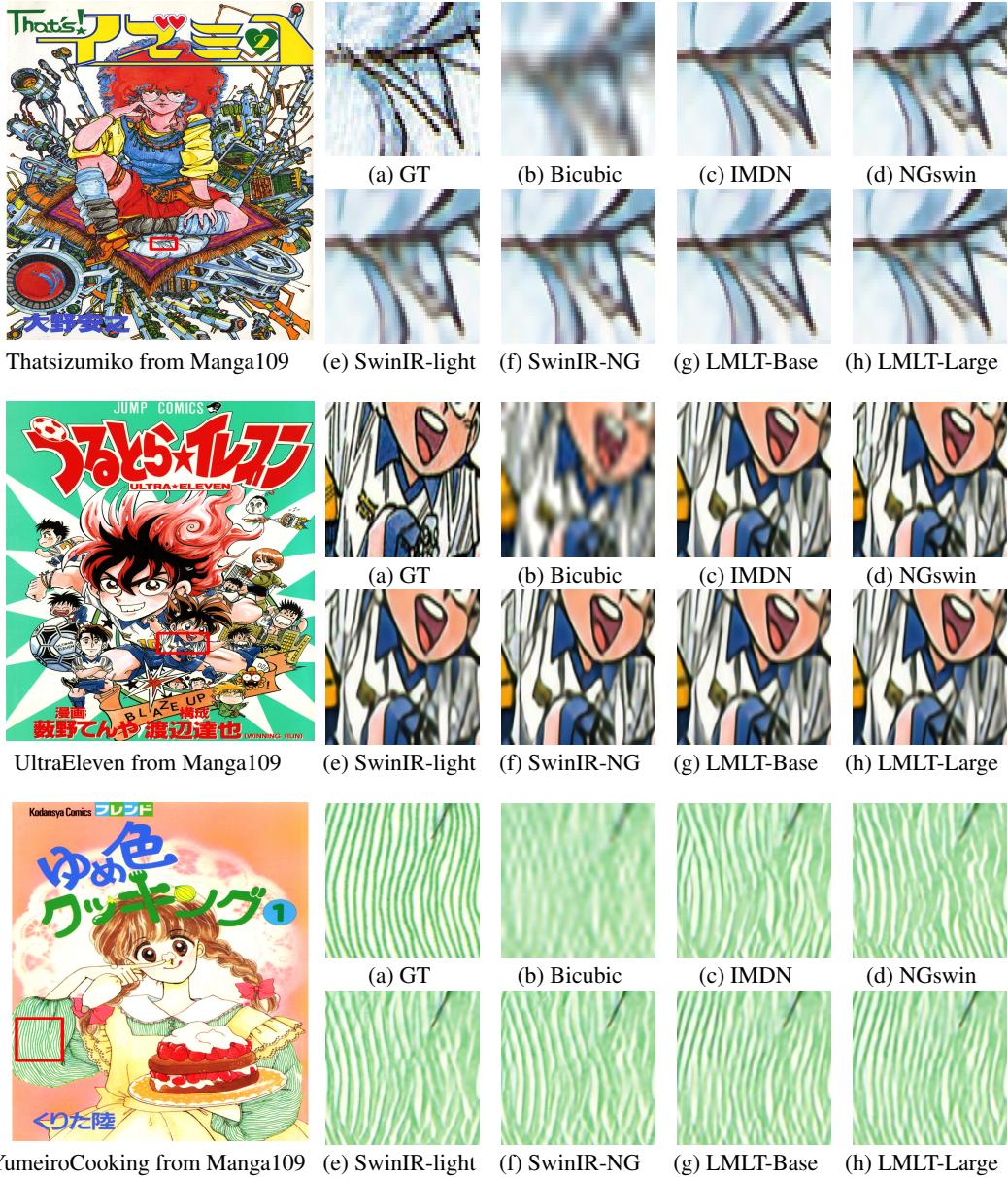


Figure H: Visual comparisons for $\times 4$ SR on Manga109 dataset. Compared with the results in (c) to (f), the Ours(LMLT-Base and LMLT-Large, (g) to (h)) restore much more accurate and clear images.