A. Outlier-Aware Loss

A.1. Analysis of Outlier-Aware Loss

We produced some histograms of the pixel-to-pixel difference values obtained by various models on several datasets, and find that often the difference values concentrate on the mean that is close to zero and show a similar frequency curve as some bell-shaped distribution with a small variance. Thus, we postulate that the true value is equal to the predicted value plus error as $y = \hat{y} + \delta$, in which δ follows an unknown distribution with a mean of 0, which is not always the case and influenced by the data distribution, loss function, and the model itself. Sometimes, the results generated by some models may have a very significant bias. But in this case, with a similar frequency curve as the Laplacian distribution due to the $\mathcal{L}_1$ loss, the losses by properly predicted pixels form the majority of the total loss when the network converges. Therefore, decreasing the losses in the majority well predicted pixels could rapidly reach a smaller total loss while ignoring the outliers. However, we think that the outliers or the badly predicted pixels as the minority outside the confidence interval shall be the focus of the optimization. Therefore, inspired by Focal Loss [63] which focuses more on the hard examples, we propose Outlier-Aware Loss $\mathcal{L}_{OA}$.

We apply the idea of Focal Loss to image restoration problems. Following that Focal Loss optimizes cross entropy, we decide to optimize $\mathcal{L}_1$ loss. Compared to $\mathcal{L}_1 = \sum_i |y_i - \hat{y}_i|$, $\mathcal{L}_2 = \sum_i (y_i - \hat{y}_i)^2$ is more sensitive to outliers as the difference is squared, which causes the suppression of high-frequency details involving the regression-to-the-mean issue [84]. Since $\mathcal{L}_2$ loss basically causes the predicted images to be blurry [67], $\mathcal{L}_1$ loss is widely utilised in image restoration networks. Thus, we decide to figure out a new loss function for the image restoration problem which shall be in the form of $\mathcal{L}_1 \times W$ where W represents the weight indicating the degree of hardness on properly predicting this pixel.

$\mathcal{L}_2$ loss assumes that data is drawn from Gaussian distribution [67], and similarly, $\mathcal{L}_1$ loss comes from the assumption that the data is drawn from Laplacian distribution as shown in Eq. 10 where μ is the mean and b is the scale parameter. In this study, we decide to apply the Laplacian distribution as the weight decay curve to rearrange the importance of the loss generated by different pixels. The loss of the terribly predicted pixels will be enlarged.

$$f(x|\mu, b) = \frac{1}{2b}\exp(-|x-\mu|/b) \tag{10}$$

The weight is adjusted to ensure $W \in [0, 1]$ and is written as $W(x|\mu, b) = 1 - \exp(-\alpha|x-\mu|/b)$. And consequently, Outlier-Aware Loss $\mathcal{L}_{OA}$ is written as Eq. 11. The parameter α could be utilised to control the degree of focus we would like to put on the accurately predicted pixels. The smaller α is, the less attention is paid to accurately predicted pixels. $\mathcal{L}_{OA}$ shall be approaching $\mathcal{L}_1$ as α gradually becomes larger, which is shown in Fig. 8. It could be observed that the number of the badly predicted outliers is reduced by $\mathcal{L}_{OA}$.

$$\mathcal{L}_{OA} = \frac{1}{HW}\sum_{i=0}^{H-1}\sum_{j=0}^{W-1}\left[|\delta_{i,j}| \times \left(1 - e^{-\alpha|\delta_{i,j}-\mu|/b}\right)\right] \tag{11}$$

The generalizability of $\mathcal{L}_{OA}$ is tested on networks as shown by Table 6. It is observed that $\mathcal{L}_{OA}$ could consistently improve the performance of those models.

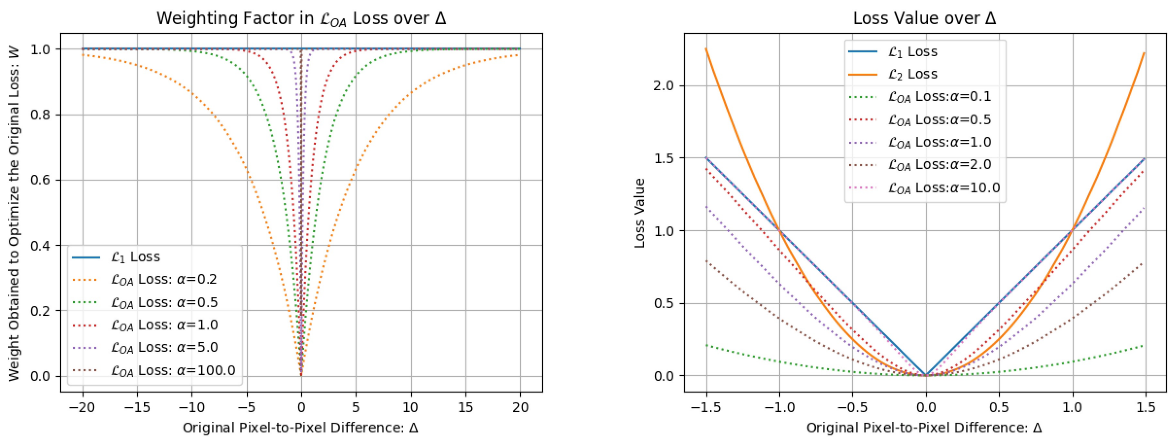
Tasks	SR				ISP			LLE		
Models	EDSR×4 [62]	EdgeSR×2 [71]	FSRCNN×2 [19]	SYENet(SR)×2	MiAigo [45]	JMU [45]	SYENet(ISP)	IAN [36]	LUM [98]	SYENet(LLE)
$\mathcal{L}_1$	28.9164	32.0771	31.8186	33.1691	23.5752	23.7805	24.7200	19.8092	19.2398	20.9717
$\mathcal{L}_{OA}$(Ours)	**28.9530**	**32.1961**	**32.1109**	**33.1837**	**23.8255**	**23.9428**	**24.8732**	**20.2487**	**19.8516**	**22.5900**

Table 6: The performance (PSNR) of different models could be consistently improved by Outlier-Aware Loss.

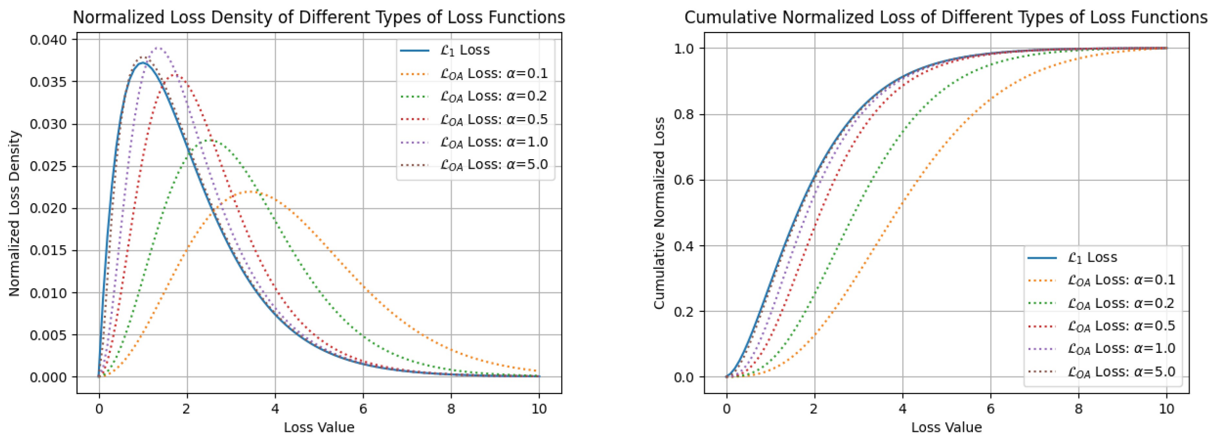
A.2. Generalized Outlier-Aware Loss

Apart from image restoration problems where $\mathcal{L}_1$ loss is mostly utilised, the Outlier-Aware Loss $\mathcal{L}_{OA}$ could be further generalized by introducing a norm parameter p and changing the definition of scale parameter b for the consistency between **MLE**(maximum likelihood estimation) and minimizing the loss function. For example, for problems where $\mathcal{L}_2$ loss outperforms $\mathcal{L}_1$ loss, we could set p to be 2 and define b to be $2\sigma^2$ for the consistency between minimizing $\mathcal{L}_2$ loss and MLE on Gaussian distribution.

$$\mathcal{L}_{OA} = \frac{1}{HW}\sum_{i=0}^{H-1}\sum_{j=0}^{W-1}\left[|\delta_{i,j}|^p \times \left(1 - e^{-\alpha|\delta_{i,j}-\mu|^p/b}\right)\right] \tag{12}$$



(a) **(left)**Weight $W(x|\mu, b) = 1 - \exp(-\alpha|x - \mu|/b)$ for adjusting the original loss over the difference between the network output and ground truth $x = \Delta = I^{(Output)} - I^{(GT)}$ as the tunable hyperparameter α varies. For simplicity, it is assumed that the mean μ and the scale parameter b of Δ are 0 and 1 respectively. It shows that larger α makes $\mathcal{L}_{OA}$ approach $\mathcal{L}_1$, while smaller α reduces the weights of the loss by well predicted pixels to a larger degree. **(right)**The loss value over Δ: compared with $\mathcal{L}_2$, $\mathcal{L}_{OA}$ reduces the weights of the well predicted pixels instead of squaring the loss values of badly predicted pixels. Since $\mathcal{L}_{OA} \leq \mathcal{L}_1$, $\mathcal{L}_1$ becomes the asymptote of $\mathcal{L}_{OA}$ as α becomes larger.



(b) **(left)**The normalized loss density that we define as loss value times the probability density of that loss value $\mathcal{L} \times p(\mathcal{L})$, which leads to the expectation of loss to be 1 as $\int_{\mathcal{L}} \mathcal{L} \times p(\mathcal{L}) d\mathcal{L} = 1$. We could understand the loss density as the contribution to the total loss by this loss value. It could be observed that $\mathcal{L}_{OA}$ reshapes the loss density by increasing the proportion of outliers and reducing the proportion of well predicted pixels in the total loss. Specifically, a smaller α causes a larger proportion transfer towards the outliers. Therefore, we successfully increase the importance of badly predicted outliers and decrease the importance of well predicted pixels in the training. Initially, we try to collect the loss values by testing the pre-trained model but found the vibration of the values cause the curve to be too messy. Thus we decide to randomly generate pixel-to-pixel loss values by sampling from the Laplacian distribution since the loss values collected from the test look very close to it. Note that this is not always the case due to the variation of the dataset and network itself. **(right)**The cumulative normalized loss, which is the integration of normalized loss density, shows the growth of total loss when we integrate the total loss from small loss values to large ones. It is observed that for $\mathcal{L}_1$ loss, the loss values smaller than 3 forms 80% of the total loss, but for $\mathcal{L}_{OA}$ loss with $\alpha = 0.1$, the loss values smaller than 3 only forms 30% of the total loss.

Figure 8: Analysis of the Outlier-Aware Loss: Combining (a) and (b) could demonstrate the function of $\mathcal{L}_{OA}$ and α. It could be seen from all the four figures that larger α make $\mathcal{L}_{OA}$ approach $\mathcal{L}_1$ but would not exceed $\mathcal{L}_1$. So that, we could avoid the regression to the mean problem by $\mathcal{L}_2$.

As mentioned above, different loss functions imply the MLE of different data distributions. And hence, a variety of loss functions should be proposed and studied to respond to the various data distributions in the real world.

B. ISP unsupervised warming up strategy

Warming up strategy used in deep learning usually aims at overcoming optimization challenges early in training [25].In our ISP task, besides this aim, we want our network to have the ability to recover the defective pixel, which often occurs in ISP raw data because of the physical issues on the camera sensor, by their neighbor pixels. Inspired by [32], we randomly mask $\frac{1}{3}$ of the input by 3×3 mask tokens and train the network to recover them with $1e-6$ learning rate for 10 epochs.

C. Re-parameterizing the ConvRep block

The input feature and output feature of ConvRep block in SYENet shall have $C^{(\mathrm{in})}$ and $C^{(\mathrm{out})}$ channels. ConvRep block has N branches, each of which can be designed specifically. Each branch could convert the channel of the input feature to $R \times C^{(\mathrm{out})}$ where R is the hyperparameter. The features by N branches shall be concatenated in channel dimension, and hence the number of channels becomes $N \times R \times C^{(\mathrm{out})}$. Then the final 1×1 convolution converts the feature with $N \times R \times C^{(\mathrm{out})}$ channels to output feature with $C^{(\mathrm{out})}$ channels. The reparameterization shall turn the complex ConvRep block back to a normal convolution block for inference.

The convolutional block injected by an input feature tensor $I^{(in)} \in \mathbb{R}^{N \times C^{(in)} \times W \times H}$ shall have a weight matrix $W \in \mathbb{R}^{C^{(out)} \times C^{(in)} \times k_W \times k_H}$ and a bias matrix $B \in \mathbb{R}^{C^{(out)}}$. In SYENet, it is set that all the convolutions generate the output with the same shape of the input feature map by arranging the padding carefully for the purpose of concatenation afterward. Batch normalization could be expressed as Eq. 13 converting input X to Y, in which γ and β are normally assigned to be 1 and 0 respectively for no linear transformation.

$$\begin{aligned} Y=&\frac{X-\mathbf{E}[X]}{\sqrt{\mathbf{VAR}[X]+\epsilon}} \times \gamma + \beta \\ =&\left(\frac{X}{\sqrt{\mathbf{VAR}[X]+\epsilon}} + \frac{-\mathbf{E}[X]}{\sqrt{\mathbf{VAR}[X]+\epsilon}}\right) \times \gamma + \beta \\ =&(W^{(\mathrm{bn})} \times X + B^{(\mathrm{bn})}) \times \gamma + \beta \end{aligned} \tag{13}$$

It is obvious that the batch normalization towards the output of the convolution layer could be converted to the batch normalization towards the weight and bias of the convolution layer, as shown in Eq. 15. In addition, the concatenation of convolution output could also be re-parameterized as the concatenation of the weights and bias of all the convolution layers, as shown by Eq. 17. In the following equations, $\bullet$ means convolution operation. It is assumed the convolution in each branch has $C^{(out)}$ output channels, and hence the concatenation of convolutions from all the branches can be reparameterized as a convolution with $N \times C^{(out)}$ output channels.

$$F^{(\mathrm{conv})} = \mathbf{CONV}(F^{(\mathrm{in})}) = F^{(\mathrm{in})} \bullet W^{(\mathrm{c})} + B^{(\mathrm{c})} \tag{14}$$

Substituting Eq. 14 into the Eq. 13 could obtain the complete expression of convolution followed by batch normalization, which is shown

$$\begin{aligned} F^{(\mathrm{conv+bn})}=&\mathbf{BN}(F^{(\mathrm{conv})}) \\ =&\big(W^{(\mathrm{bn})} \times F^{(\mathrm{conv})} + B^{(\mathrm{bn})}\big) \times \gamma + \beta \\ =&\big[W^{(\mathrm{bn})}(F^{(\mathrm{in})} \bullet W^{(\mathrm{c})} + B^{(\mathrm{c})}) + B^{(\mathrm{bn})}\big] \times \gamma + \beta \\ =&(F^{(\mathrm{in})} \bullet W^{(\mathrm{bn})} W^{(\mathrm{c})} + W^{(\mathrm{bn})} B^{(\mathrm{c})} + B^{(\mathrm{bn})}) \times \gamma + \beta \\ =&F^{(\mathrm{in})} \bullet \gamma W^{(\mathrm{bn})} W^{(\mathrm{c})} + \gamma (W^{(\mathrm{bn})} B^{(\mathrm{c})} + B^{(\mathrm{bn})}) + \beta \\ =&F^{(\mathrm{in})} \bullet W^{(\mathrm{c+bn})} + B^{(\mathrm{c+bn})} \end{aligned} \tag{15}$$

$$
\begin{aligned}
F^{(\mathrm{cat})} =& \mathbf{CAT}(F^{(\mathrm{conv})}, F^{(\mathrm{conv+bn})}) \\
=& F^{(\mathrm{in})} \bullet \mathbf{CAT}(W^{(\mathrm{c+bn})}) + \mathbf{CAT}(B^{(\mathrm{c+bn})}) \\
=& F^{(\mathrm{in})} \bullet W^{(\mathrm{cat})} + B^{(\mathrm{cat})}
\end{aligned} \tag{16}
$$

The numbers of $F^{(\mathrm{conv})}$ and $F^{(\mathrm{conv+bn})}$ to be concatenated may vary according to the specific implementation and could be 0 if necessary.

$$
\begin{aligned}
F^{(\mathrm{rep})} =& \mathbf{CONV}_{1\times1}(F^{(\mathrm{cat})}) \\
=& (F^{(\mathrm{in})} \bullet W^{(\mathrm{cat})} + B^{(\mathrm{cat})}) \bullet W^{(\mathrm{c})} + B^{(\mathrm{c})} \\
=& F^{(\mathrm{in})} \bullet \mathbf{MATMUL}(W^{(\mathrm{cat})}, W^{(\mathrm{c})}) + \mathbf{MATMUL}(B^{(\mathrm{cat})}, W^{(\mathrm{c})}) + B^{(\mathrm{c})} \\
=& F^{(\mathrm{in})} \bullet W^{(\mathrm{rep})} + B^{(\mathrm{rep})}
\end{aligned} \tag{17}
$$

MATMUL means matrix multiplication. $W^{(\mathrm{c})}$ and $W^{(\mathrm{c+bn})}$ both have the shape of $C^{(\mathrm{out})} \times C^{(\mathrm{in})} \times K_H \times K_W$, while $W^{(\mathrm{c+cat})}$ after concatenation has the shape of $NRC^{(\mathrm{out})} \times C^{(\mathrm{IN})} \times K_H \times K_W$ where N is the number of branches and K_H and K_W are the kernel size. As the input feature and output feature of this ConvRep block are designed to have $C^{(\mathrm{in})}$ and $C^{(\mathrm{out})}$ channels, the final $\mathrm{CONV}_{1\times1}$ after concatenation should have weight matrix with the shape of $C^{(\mathrm{in})} \times NRC^{(\mathrm{out})} \times 1 \times 1$. By squeezing, permuting, and matrix multiplication operations, the reparameterized weight matrix $W^{(\mathrm{rep})}$ and bias matrix $B^{(\mathrm{rep})}$ could be obtained as shown by Eq. 17.

D. Two-branch feature fusion

After the re-parameterization, the two-branch structure could be converted back to two parallel convolution blocks and later on be fused by a multiplication operation as activation. The output fusion by **QCU** is represented by Eq. 18, and each branch can be represented as equations below Eq.19 and Eq. 20 as $\hat{I}$ and $\tilde{I}$. $B \in \mathbb{R}^{N\times C\times1\times1}$ is the learnable bias in **QCU**.

$$
I = \hat{I} \times \tilde{I} + B \tag{18}
$$

For the convolution with kernel size of (k_m, k_n), $k_m^{(h)}$ and $k_n^{(h)}$ are defined as $\frac{1}{2}(k_m - 1)$ and $\frac{1}{2}(k_n - 1)$. l is the index of channels for input and output, while L represents the total number of channels.

$$
\hat{I}_{i,j}^{l_{out}} = \hat{b}^{l_{out}} + \sum_{\hat{l}_{in}=0}^{L_{in}} \sum_{\hat{m}=-k_m^{(h)}}^{k_m^{(h)}} \sum_{\hat{n}=-k_n^{(h)}}^{k_n^{(h)}} \left(\hat{I}_{i+\hat{m},j+\hat{n}}^{l_{in}} \times \hat{K}_{\hat{m},\hat{n}}^{l_{out},l_{in}}\right) \tag{19}
$$

$$
\tilde{I}_{i,j}^{l_{out}} = \tilde{b}^{l_{out}} + \sum_{\tilde{l}_{in}=0}^{L_{in}} \sum_{\tilde{m}=-k_m^{(h)}}^{k_m^{(h)}} \sum_{\tilde{n}=-k_n^{(h)}}^{k_n^{(h)}} \left(\tilde{I}_{i+\tilde{m},j+\tilde{n}}^{l_{in}} \times \tilde{K}_{\tilde{m},\tilde{n}}^{l_{out},l_{in}}\right) \tag{20}
$$

So that, the output feature value for a certain output channel in a fixed position (i, j) can be represented precisely as Eq. 21 which could be further expressed as Eq. 22.

$$
I_{i,j}^{l_{out}} = \hat{I}_{i,j}^{l_{out}} \times \tilde{I}_{i,j}^{l_{out}} + b_{i,j} \tag{21}
$$

$$
\begin{aligned}
I_{i,j}^{l_{out}} =& b_{i,j} + \hat{b}^{l_{out}}\tilde{b}^{l_{out}} + \sum_{l_{in}=0}^{L_{in}} \sum_{m=-k_m^{(h)}}^{k_m^{(h)}} \sum_{n=-k_n^{(h)}}^{k_n^{(h)}} \left(\hat{I}_{i,j}^{l_{out}} \hat{K}_{\hat{m},\hat{n}}^{l_{out},l_{in}} \tilde{b}^{l_{out}} + \tilde{I}_{i,j}^{l_{out}} \tilde{K}_{\tilde{m},\tilde{n}}^{l_{out},l_{in}} \hat{b}^{l_{out}}\right) + \\
& \sum_{\tilde{l}_{in}=0}^{L_{in}} \sum_{\hat{l}_{in}=0}^{L_{in}} \sum_{\tilde{m}=-k_m^{(h)}}^{k_m^{(h)}} \sum_{\hat{m}=-k_m^{(h)}}^{k_m^{(h)}} \sum_{\tilde{n}=-k_n^{(h)}}^{k_n^{(h)}} \sum_{\hat{n}=-k_n^{(h)}}^{k_n^{(h)}} \left(I_{i+\tilde{m},j+\tilde{n}}^{\tilde{l}_{in}} \times K_{\tilde{m},\tilde{n}}^{l_{out}\tilde{l}_{in}} \times I_{i+\hat{m},j+\hat{n}}^{\hat{l}_{in}} \times K_{\hat{m},\hat{n}}^{l_{out}\hat{l}_{in}}\right)
\end{aligned} \tag{22}
$$

Consequently, the general formulation of the output feature expression obtained by the re-parameterized two-branch structure shall be $B + (\hat{K} + \tilde{K})I + KI^2$ which is in quadratic form rather than the linear form of $B + KI$. So, the capability of feature representation by this structure could be enhanced.

However, the expression that could be converted into the form of $(\hat{K}\hat{I} + \hat{B}) \times (\tilde{K}\tilde{I} + \tilde{B})$ have a strict constraint in feature representation pattern, which means the representation must have two sets of fixed values in solution space that are $(-\hat{I}/\hat{B}, 0)$ and $(-\tilde{I}/\tilde{B}, 0)$. Hence in order to solve this issue, an additional learnable bias B is merged into the network to boost the ability to learn features.

E. Theoretical analysis about different fusion methods on a toy example

As mentioned in the paper, we compared different fusion methods including ADD(element-wise addition), CAT+CONV(concatenation plus convolution), MUL(element-wise multiplication), and the proposed QCU(Quadratic Connection Unit). As presented by Fig. 4 and Table 4, QCU gains higher PSNR than these techniques. It could be roughly explained by that the mathematical expression of QCU shows more powerful representative capability than other fusion methods.

The analysis is conducted on a toy example, which is a simple two-branch network with input tensor X of shape $\mathrm{N} \times \mathrm{C} \times \mathrm{H} \times \mathrm{W}$, output tensor Y, and two convolutions in each branch $\mathrm{CONV}^{(\mathrm{c})}$ and $\mathrm{CONV}^{(\mathrm{s})}$. Assuming the input channels and output channels are all equal to C, and consequently, we have the weights and biases of the two convolutions, which are $W^{(\mathrm{c})}$, $B^{(\mathrm{c})}$, $W^{(\mathrm{s})}$ and $B^{(\mathrm{s})}$, all with C channels. For CAT+CONV method, we have one extra convolution $\mathrm{CONV}^{(\mathrm{cat})}$ with input channel 2C, output channel C, weight $W^{(\mathrm{cat})} = [W^{(\mathrm{catc})}, W^{(\mathrm{cats})}]$ and bias $B^{(\mathrm{cat})} = [B^{(\mathrm{catc})}, B^{(\mathrm{cats})}]$. $W^{(\mathrm{cat})}$ has shape $\in \mathrm{C} \times 2\mathrm{C} \times \mathrm{K} \times \mathrm{K}$, while $W^{(\mathrm{catc})}$ and $W^{(\mathrm{cats})}$ have shape $\in \mathrm{C} \times \mathrm{C} \times \mathrm{K} \times \mathrm{K}$, in which K is the kernel size.

The model could be further simplified by setting $\mathrm{C} = 1$, and kernel size to be 1 as well. The channel and kernel assumption could convert the convolution into simple linear transformation.

$$\begin{aligned} Y^{(\mathrm{ADD})} &= (W^{(\mathrm{c})}X + B^{(\mathrm{c})}) + (W^{(\mathrm{s})}X + B^{(\mathrm{s})}) \\ &= (W^{(\mathrm{c})} + W^{(\mathrm{s})})X + (B^{(\mathrm{c})} + B^{(\mathrm{s})}) \end{aligned} \tag{23}$$

$$Y^{(\mathrm{CAT})} = \mathbf{CAT}(W^{(\mathrm{c})}X + B^{(\mathrm{c})}, W^{(\mathrm{s})}X + B^{(\mathrm{s})}) \tag{24}$$

$$\begin{aligned} Y^{(\mathrm{CAT+CONV})} &= W^{(\mathrm{cat})}Y^{(\mathrm{CAT})} + B^{(\mathrm{cat})} \\ &= W^{(\mathrm{catc})}(W^{(\mathrm{c})}X + B^{(\mathrm{c})}) + B^{(\mathrm{catc})} + W^{(\mathrm{cats})}(W^{(\mathrm{s})}X + B^{(\mathrm{s})}) + B^{(\mathrm{cats})} \\ &= (W^{(\mathrm{catc})}W^{(\mathrm{c})} + W^{(\mathrm{cats})}W^{(\mathrm{s})})X + (W^{(\mathrm{catc})}B^{(\mathrm{c})} + B^{(\mathrm{catc})} + W^{(\mathrm{cats})}B^{(\mathrm{s})} + B^{(\mathrm{cats})}) \end{aligned} \tag{25}$$

$$\begin{aligned} Y^{(\mathrm{MUL})} &= (W^{(\mathrm{c})}X + B^{(\mathrm{c})}) \times (W^{(\mathrm{s})}X + B^{(\mathrm{s})}) \\ &= W^{(\mathrm{c})}W^{(\mathrm{s})}X^2 + (W^{(\mathrm{c})}B^{(\mathrm{s})} + W^{(\mathrm{s})}B^{(\mathrm{c})})X + B^{(\mathrm{c})}B^{(\mathrm{s})} \end{aligned} \tag{26}$$

$$\begin{aligned} Y^{(\mathrm{QCU})} &= (W^{(\mathrm{c})}X + B^{(\mathrm{c})}) \times (W^{(\mathrm{s})}X + B^{(\mathrm{s})}) + B^{(\mathrm{QCU})} \\ &= W^{(\mathrm{c})}W^{(\mathrm{s})}X^2 + (W^{(\mathrm{c})}B^{(\mathrm{s})} + W^{(\mathrm{s})}B^{(\mathrm{c})})X + (B^{(\mathrm{c})}B^{(\mathrm{s})} + B^{(\mathrm{QCU})}) \end{aligned} \tag{27}$$

Comparing the expressions of all the fusion methods, the representative capability of these methods could be summarized. As indicated by Table 4, the performance on PSNR by the fusion methods is ranked as QCU $>$ MUL $>$ CAT+CONV $>$ ADD. It is observed that QCU and MUL both have an extra second-order term X^2, which makes QCU and MUL have the more powerful representative capability to fit more complex models. On the other hand, CAT+CONV could be regarded as a weighted addition operation, in which the weights of $W^{(\mathrm{c})}$, $W^{(\mathrm{s})}$, $B^{(\mathrm{c})}$, and $B^{(\mathrm{s})}$ are all adjusted by the following convolution $\mathbf{CONV}^{(\mathrm{cat})}$. Thus, CAT+CONV fusion could generate more complex fitting. It should be noted that those tricks could significantly improve the performance of small networks. But in large models, since the networks already have very large depth and width to fit complex distributions, those tricks might not be necessarily helpful.

F. Comparisons with models in SR task with scale factor $\times 3$

More comparisons for the scale factor of $\times 3$ are shown in Table 7. We trained these models on DIV2K and tested them on Set5, Set14, BSD100, and Urban100 following Table 1.

Method	Scale	#P	Avg latency(ms)	FPS(2K)	Set5	Set14	BSD100	BSD100 Score	Urban100	Urban100 Score
ABPN [20]	$\times 3$	53K	59.3	17	32.99	29.46	28.48	2.264	26.34	2.434
SCSRN [42]	$\times 3$	67K	43.1	23	**33.20**	**29.57**	**28.56**	3.634	**26.52**	4.294
HOPN [42]	$\times 3$	48K	32.1	31	32.66	29.28	28.35	3.647	26.33	2.966
SYENet (**Ours**)	$\times 3$	7.9K	**11.3**	88	32.93	29.44	28.47	**12.234**	26.33	**12.595**

Table 7: Comparison on super-resolution issue between the results by PSNR(dB), SSIM, and comprehensive score with SOTA: the models are ranked by score defined by Eq. 9 on BSD100 dataset. The normalization factors C for BSD100 Score are $1.8\text{E}16(\times 2)$, $1\text{E}14(\times 3)$, and $5\text{E}14(\times 4)$. While the normalization factors C for Urban100 Score are $2.5\text{E}15(\times 2)$, $3.5\text{E}12(\times 3)$, and $1.6\text{E}13(\times 4)$.

G. Constraints of low-level vision tasks in mobile devices

G.1. Constraints of input size and output size

Low-level vision tasks are basically regression questions. The output of the model is at least the same size as the input, and particularly for super-resolution problems, the output size shall be many times larger than that of the input. However, the output of high-level vision tasks like classification questions could be a vector containing scores of different object labels, which is much smaller than the low-level vision tasks output. Apart from output size, the input to high-level vision tasks could also be significantly smaller than low-level vision task input. Normally, the input image would be resized into as small as 128×128 or 256×256. However, for low-level vision tasks, the information of pixels is rather vital and should not be down-scaled or degraded before processing.

Methods	Task	$\#P$	Input	Output	MACs(G)	FLOPs(G)	Latency(Platform)
EfficientFormer [59]	Classification	37.1M	224×224	-	3.57	-	4.2ms(iPhone 12 NPU)
LightViT-B [38]	Classification	35.2M	224×224	-	-	3.9	1.2ms(V100 GPU)
SYENet(Ours)	SR×2	4.9K	960×540	1920×1080	-	2.83	16.5ms(Qualcomm Snapdragon mobile SoC)

Table 8: It is obviously shown from the comparison between SYENet and efficient networks for high-level vision task that low-level vision task is more challenging as it requires much fewer parameters to achieve real-time inference due to the large input and output size.

Two typical examples of efficient high-level vision networks are EfficientFormer [59] and LightViT [38]. With 37.1 million parameters, EfficientFormer Supernet could still achieve an extremely low latency of 4.2ms on iPhone 12 NPU. While, as shown in Table 8, our proposed method with only 4.9k parameters could only reach a latency of 16.5ms on Qualcomm Snapdragon mobile SoC due to the large input and output size.

G.2. Constraints of small memory bandwidth of mobile devices

Information Multi-distillation Block IMDB [39] is widely utilised in the efficient SR challenges [50] by many participators as well as the winning teams in some years. However, the structure of IMDB involves a serious disadvantage for mobile devices as it requires the network to store the results generated by each step which should be concatenated after all the steps are done. Therefore, during the inference, once the memory is too small, and consequently data transfer between the memory and disk occurs, the latency shall be seriously enlarged. Thus, keeping memory usage small is another challenge for low-level vision tasks. This is also the reason that the structure of IMDB is not employed in SYENet.

G.3. The influence of network depth towards network latency

The tradeoff between the network depth and kernel size is shown in Table 9. Apparently, one 5×5 convolution requires more numerical calculations than two 3×3 convolutions. However, as indicated by Table 9, deeper networks have more data transfer operations between DRAM and local SRAM when the feature map size is too huge. Thus, the larger depth will also enlarge the latency as well. Sometimes, the influence of depth could be even larger than FLOPs in certain platforms due to the inconsistency between hardware acceleration policies and network structures. Therefore, taking the hardware acceleration policies into consideration is also vital for designing the network structure.

Methods	PSNR	Latency(ms)	MAIISP Score
$\mathrm{CONV}_{5\times5}$	24.8732	**11.4**	**16.57**
$\mathrm{CONV}_{3\times3} \rightarrow \mathrm{CONV}_{3\times3}$	24.8824	12.9	14.83
$\mathrm{CONV}_{3\times3} \rightarrow \mathrm{PReLU} \rightarrow \mathrm{CONV}_{3\times3}$	**24.9349**	13.4	15.35

Table 9: Depth-kernel size tradeoff: tested on MAI ISP dataset

H. Disadvantage of SYENet on VSR task

The experiment results of SYENet on Video SR problems are very unsatisfactory, which is even worse than that of SISR. As suggested by BasicVSR [6], the VSR network should have four modules which are propagation, alignment, aggregation, and upsampling. Since SYENet is too small to implement the propagation and alignment module, SYENet could not utilise the information of neighboring frames to enhance the VSR performance. In fact, the neighboring frames injected into SYENet actually impair the restoration as they provide misaligned pixel information that SYENet cannot make good use of.

I. More low-light enhancement comparisons

The images are shown in Fig. 9.

J. More image signal processing comparisons

The images are shown in Fig. 10.

K. More Feature maps by each branch

The images are shown in Fig. 11.

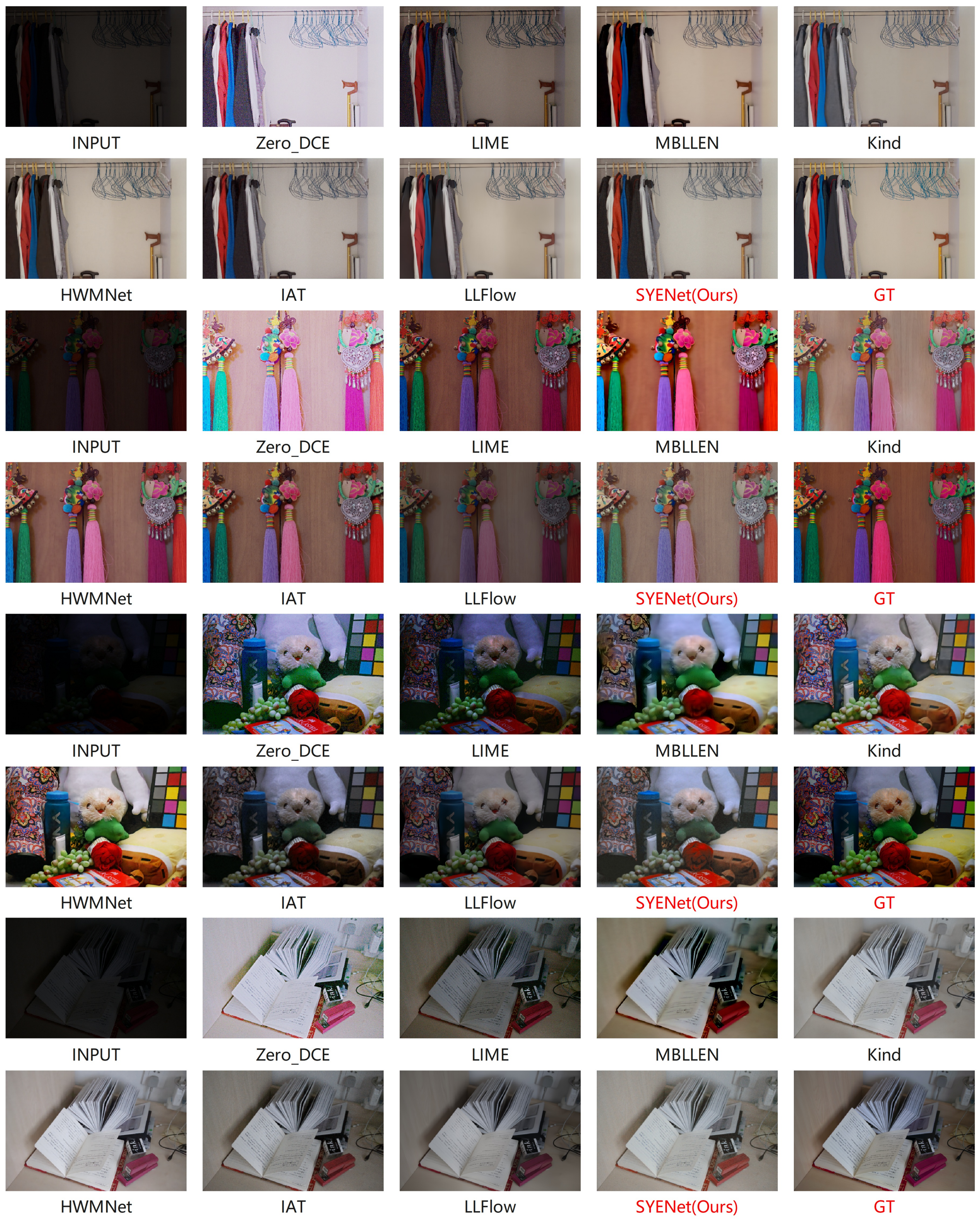


Figure 9: Low-light enhancement comparisons

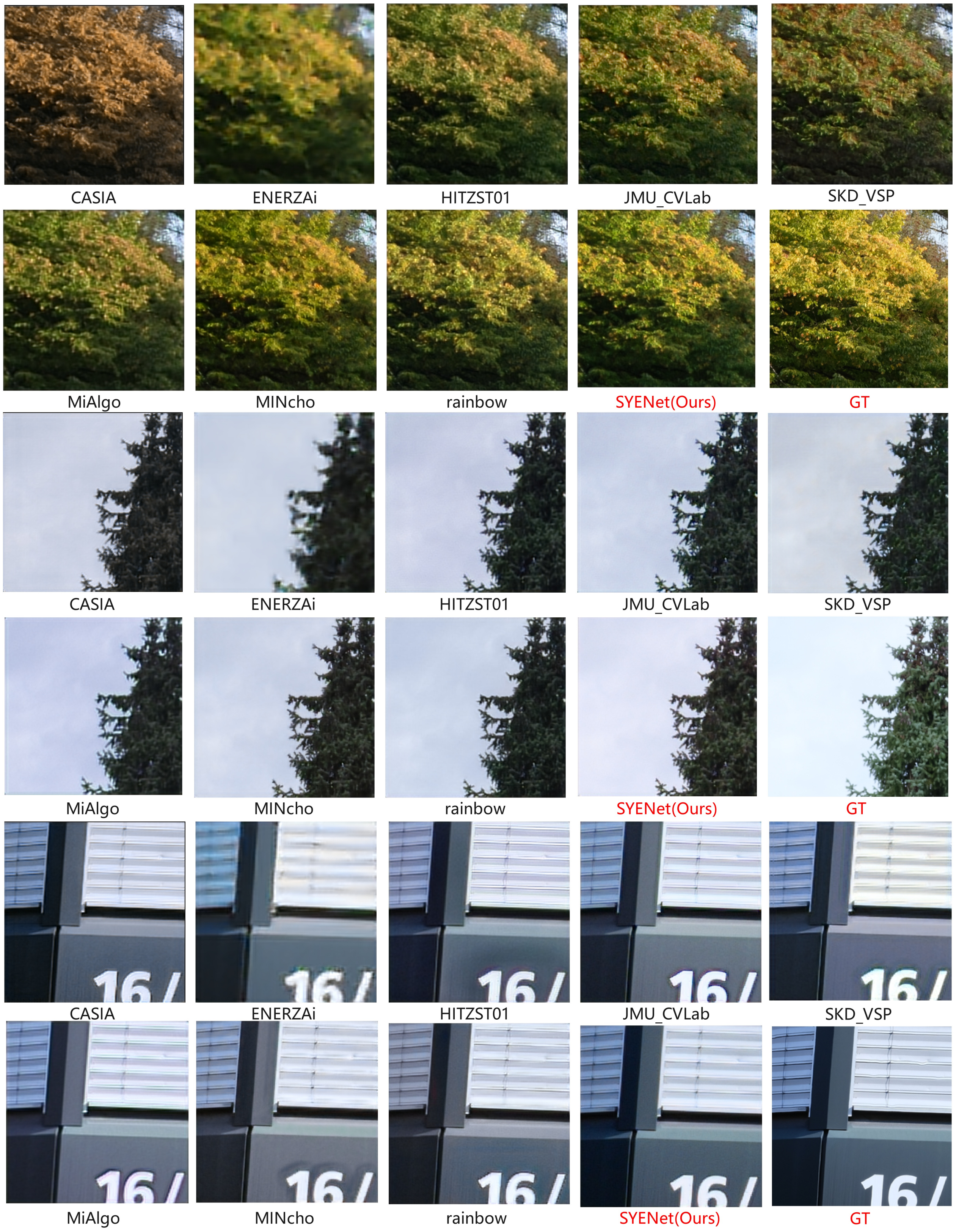


Figure 10: Image signal processing comparisons

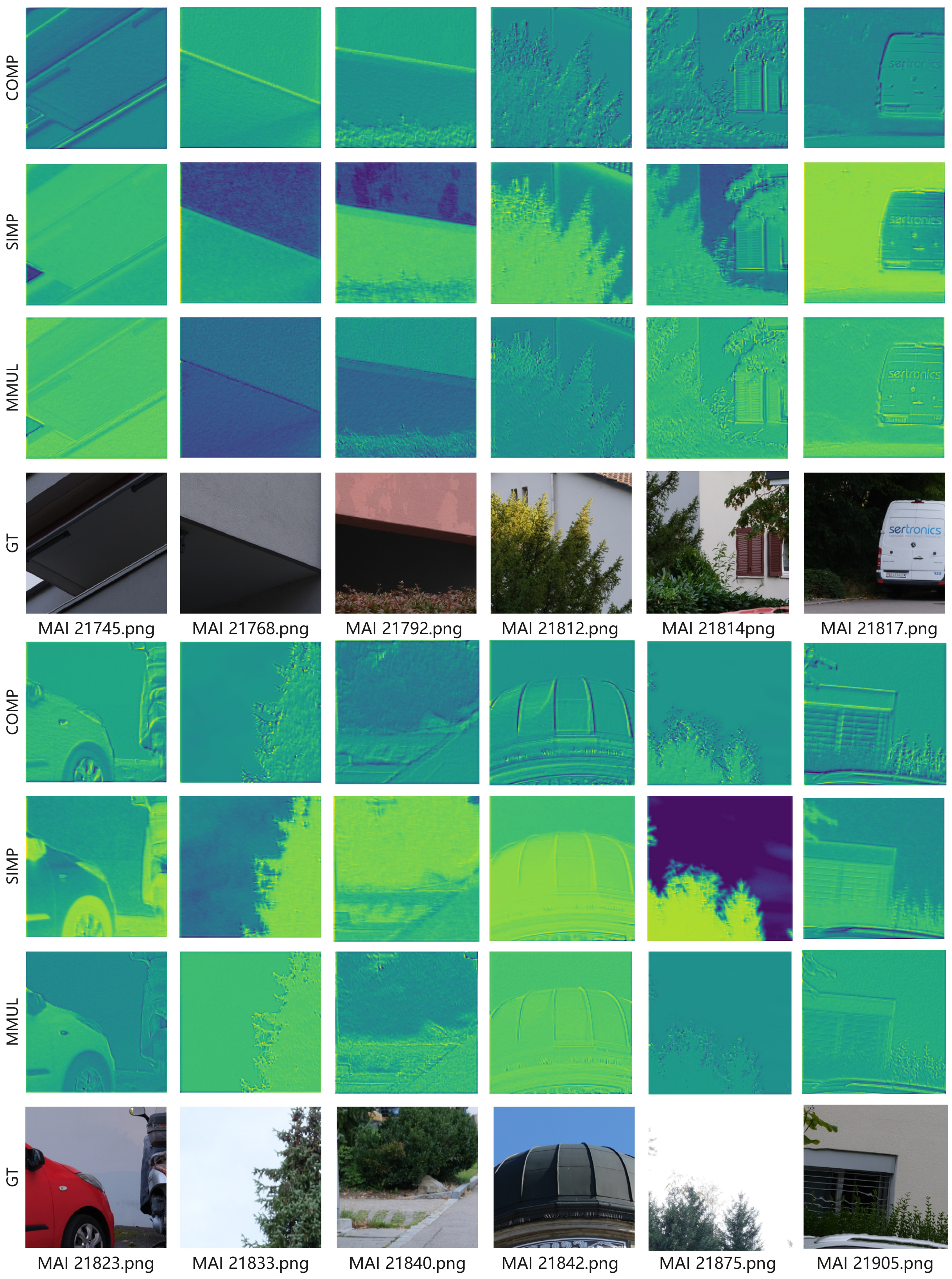


Figure 11: Feature maps of each branch and fused result