EnsIR: An Ensemble Algorithm for Image Restoration via Gaussian Mixture Models

Shangquan Sun[1,2], Wenqi Ren[3], Zikun Liu[4], Hyunhee Park[4], Rui Wang[1,2], Xiaochun Cao[3]

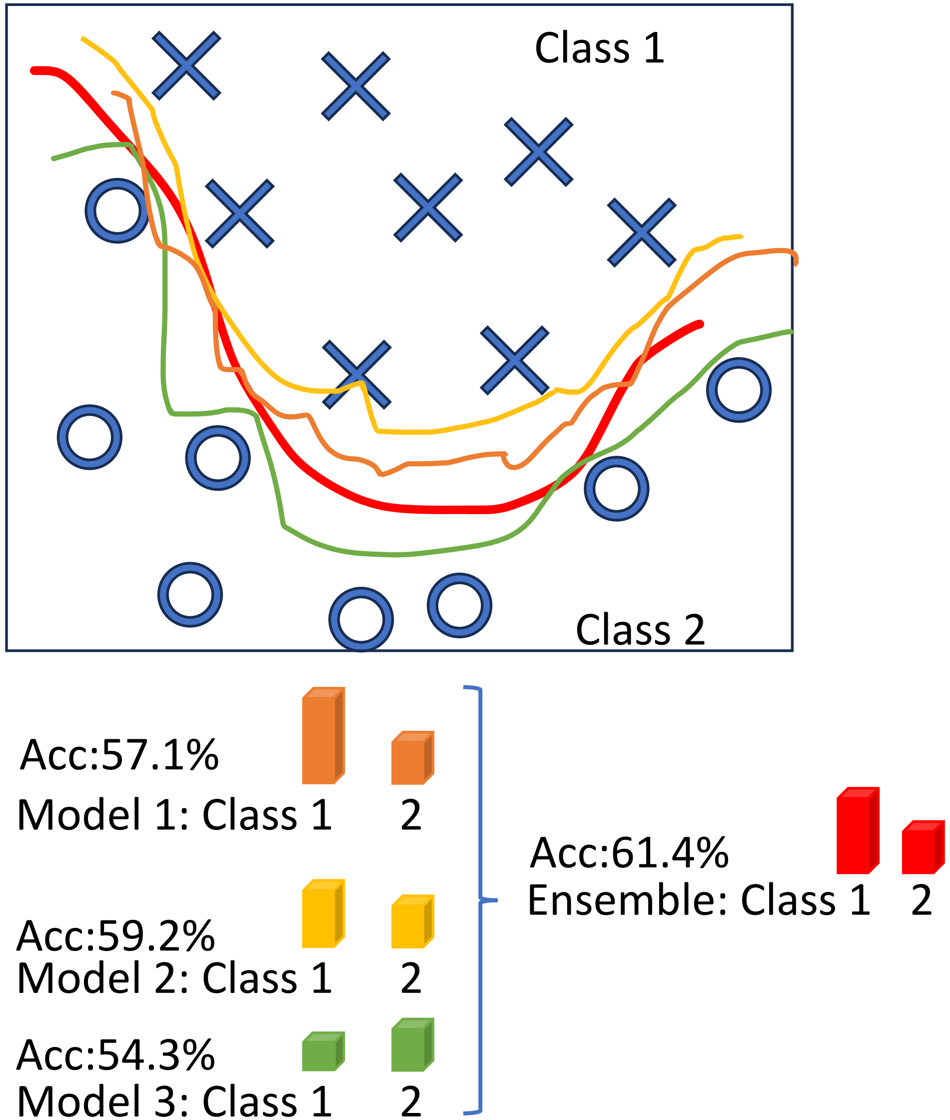
[1]CAS, China [2]UCAS, China [3]Shenzhen Campus of SYSU, China [4]Samsung

NeurIPS 2024

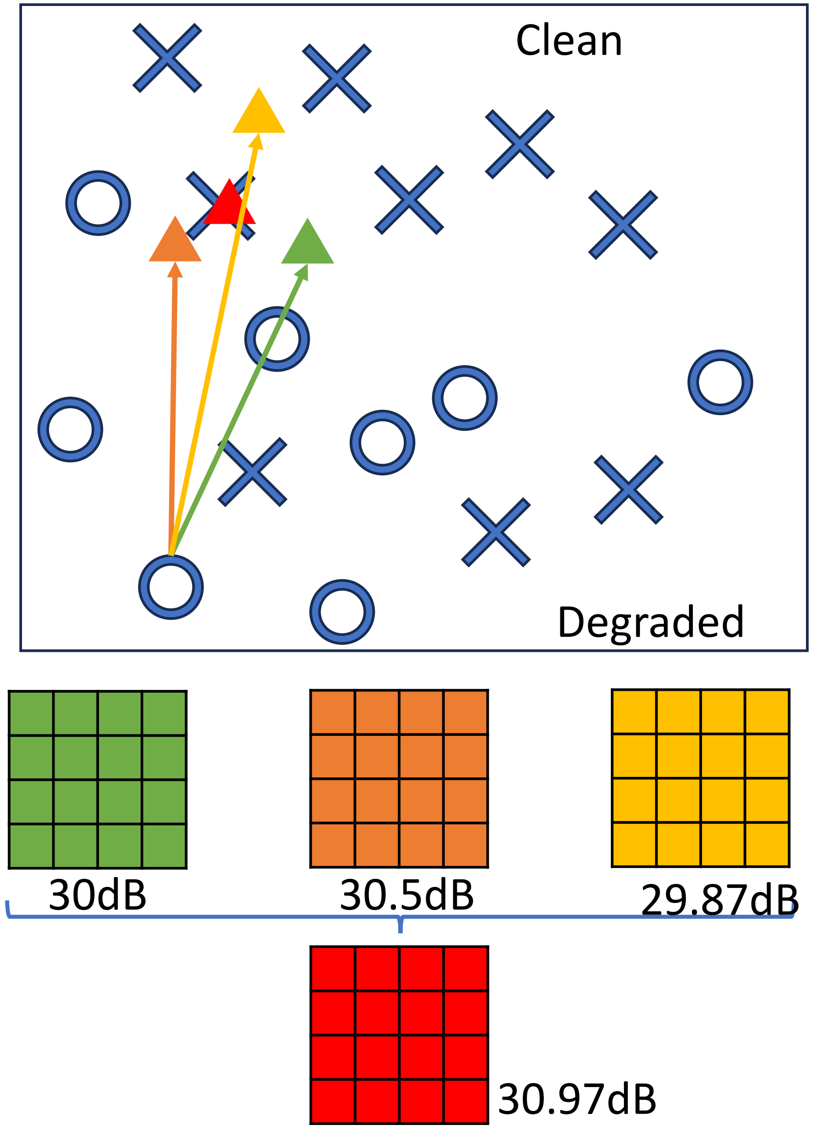
2024.11.2

Motivation

Classification: $\mathbb{R}^{3\times H\times W} \rightarrow K$



Restoration: $\mathbb{R}^{3\times H\times W} \rightarrow \mathbb{R}^{3\times H\times W}$



Problem Formulation

- Ensemble for Image Restoration

$$\tilde{\mathbf{Y}}_n = \boldsymbol{\beta}_n^\top \left[\tilde{\mathbf{X}}_{1,n} \quad \cdots \quad \tilde{\mathbf{X}}_{M,n}\right], \forall n \tag{1}$$

- For a reference set, formulate all data with Gaussian prior

$$\mathbf{y}_{1:N} | f_m, \mathbf{x}_{1:N} \sim \mathcal{N}\left(\mathbf{x}_{m,1:N}, \mathrm{diag}(\boldsymbol{\Sigma}_{m,1}, ..., \boldsymbol{\Sigma}_{m,N})\right), \tag{2}$$

- Partition data into bins:

$$\mathbf{y}_{r,1:N} = \mathbf{R}_r \cdot \mathbf{y}_{1:N}, \qquad \mathbf{x}_{r,m,1:N} = \mathbf{R}_r \cdot \mathbf{x}_{m,1:N}, \tag{3}$$

- Within each bin, Gaussian prior still holds

$$\mathbf{y}_{r,1:N}^{(i)} | f_m, \mathbf{x}_{r,m,1:N} \overset{i.i.d}{\sim} \mathcal{N}(\mu_{r,m,1:N}, \sigma_{r,m,1:N}), \ \forall i \in [1, ..., N_r], \tag{4}$$

- Then ensemble problem is split for estimating weights within. bins

$$\mathbf{y}_{r,1:N}^{(i)} = \mathbb{E}_z\left[\mathbf{x}_{r,m,1:N}^{(i)}\right] = \sum_{m=1}^{M} \alpha_{r,m} \cdot \mathbf{x}_{r,m,1:N}^{(i)}; \ P(\mathbf{y}_{r,1:N}^{(i)}) = \sum_{m=1}^{M} \alpha_{r,m} P\left(\mathbf{y}_{r,1:N}^{(i)} \middle| z = m\right), \tag{5}$$

$$\{\alpha_{r,m}\}_{r,m} \in \arg\max P(\mathbf{y}_{1:N}). \tag{6}$$

Gaussian Mixture Models and EM Algorithm

- Gaussian Mixture Models:

$$\alpha_{r,m} \in \arg\max_{\alpha_{r,m}} P(\mathbf{y}_{r,1:N}) = \arg\max_{\alpha_{r,m}} \prod_{i=1}^{N_r} P(\mathbf{y}_{r,1:N}^{(i)}). \tag{7}$$

- Estimate ensemble weights by maximizing the log likelihood as

$$\begin{aligned}
\log P(\mathbf{y}_{r,1:N}) &= \log \prod_{i=1}^{N_r} \sum_{m=1}^{M} \alpha_{r,m} \phi \left(\mathbf{y}_{r,1:N}^{(i)}; \mu_{r,m,1:N}, \sigma_{r,m,1:N} \right) \\
&= \sum_{i=1}^{N_r} \log \sum_{m=1}^{M} \alpha_{r,m} \phi \left(\mathbf{y}_{r,1:N}^{(i)}; \mu_{r,m,1:N}, \sigma_{r,m,1:N} \right) \\
&\geq \sum_{m=1}^{M} P\left(z = m \middle| \mathbf{y}_{r,1:N}^{(i)} \right) \log \frac{\alpha_{r,m} \phi(\mathbf{y}_{r,1:N}^{(i)}; \mu_{r,m,1:N}, \sigma_{r,m,1:N})}{P\left(z = m \middle| \mathbf{y}_{r,1:N}^{(i)} \right)},
\end{aligned} \tag{8}$$

Gaussian Mixture Models and **EM Algorithm**

- **E**xpectation step:

$$\gamma_{r,m,1:N} \leftarrow P\left(z = m \middle| \mathbf{y}^{(i)}_{r,1:N}\right) = \frac{\alpha_{r,m}\phi\left(\mathbf{y}^{(i)}_{r,1:N}; \mu_{r,m,1:N}, \sigma_{r,m,1:N}\right)}{\sum_{m=1}^{M} \alpha_{r,m}\phi\left(\mathbf{y}^{(i)}_{r,1:N}; \mu_{r,m,1:N}, \sigma_{r,m,1:N}\right)}. \tag{9}$$

- **M**aximization step:

$$\alpha_{r,m} \leftarrow \frac{1}{N_r}\sum_{i=1}^{N_r} \gamma_{r,m,1:N} \tag{10}$$

$$\sigma_{r,m,1:N} \leftarrow \frac{\sum_{i=1}^{N_r} \gamma_{r,m,1:N}\left(\mathbf{y}^{(i)}_{r,m,1:N} - \mu_{r,m,1:N}\right)^2}{\sum_{n=1}^{N_r} \gamma_{r,m,1:N}}. \tag{11}$$

- Priors of Mean and Variance:

$$\mu_{r,m,1:N} \leftarrow \frac{1}{N_r}\sum_{i=1}^{N_r} \mathbf{x}^{(i)}_{r,m,1:N} \tag{12}$$

$$\sigma_{r,m,1:N} \leftarrow \frac{1}{N_r}\left\|\mathbf{x}_{r,m,1:N} - \mu_{r,m,1:N}\right\|_2. \tag{13}$$

Look-up Table for inference and Overall Algorithm

- LUT to save the weights for inference

$$\tilde{\mathbf{X}}_{r,m,n} = \mathbf{R}_r \cdot \tilde{\mathbf{X}}_{m,n}, \text{ where } \mathbf{R}_r = \prod_{m=1}^{M} \mathbf{I}_{B_m}(\tilde{\mathbf{X}}_{m,n}). \tag{14}$$

$$\tilde{\mathbf{Y}}_n = \sum_{r=1}^{T^M} \tilde{\mathbf{Y}}_{r,n} = \sum_{r=1}^{T^M} \sum_{m=1}^{M} \alpha_{r,m} \tilde{\mathbf{X}}_{r,m,n}. \tag{15}$$

Algorithm 2: MPEM: EM algorithm with known Mean Prior

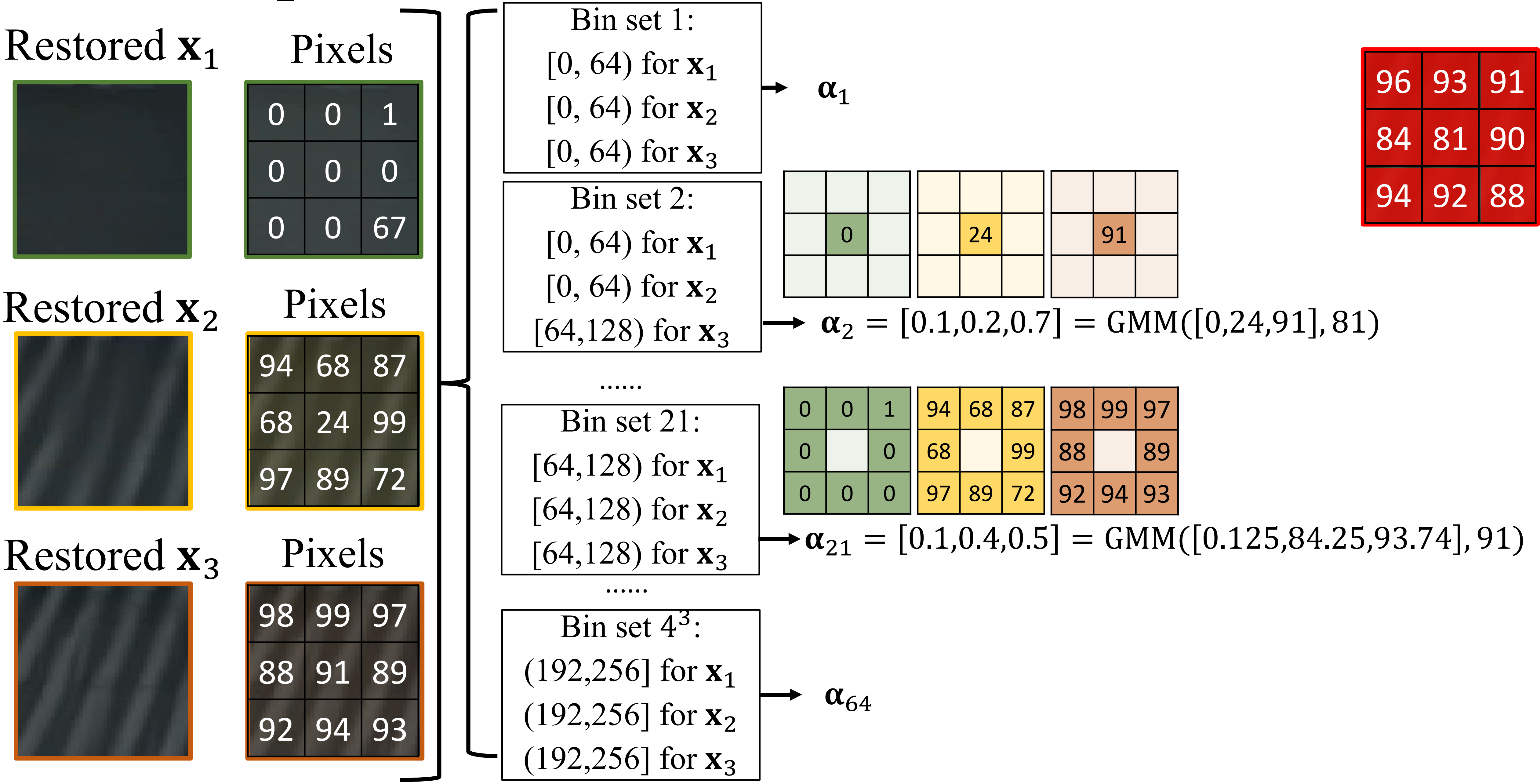
Input: $\mathbf{y}_{r,1:N}, \mathbf{x}_{r,1,1:N}, ..., \mathbf{x}_{r,M,1:N}$
Output: $(\alpha_{r,1}, ..., \alpha_{r,M})$

$N_r \leftarrow$ number of nonzero pixels in $\mathbf{y}_{r,1:N}$;
Initialize $\mu_{r,m,1:N}$ by Eq. 12;
Initialize $\sigma_{r,m,1:N}$ by Eq. 13;
while *not converge* **do**
  for $i \in [1, N_r]$ **do**
    for $m \in [1, M]$ **do**
      Update $\gamma_{r,m,1:N}$ by Eq. 9;
    end
  end
  for $m \in [1, M]$ **do**
    Update $\alpha_{r,m}$ by Eq. 10;
    Update $\sigma_{r,m,1:N}$ by Eq. 11;
  end
end

Algorithm 1: EnsIR: an ensemble algorithm for image restoration

Input: A small reference dataset $\{\mathbf{x}_n, \mathbf{y}_n\}_{n=1}^{N}$ for ensemble weight estimation, test set $\{\hat{\mathbf{X}}_n\}$, M pre-trained models $f_1, ..., f_M$, bin width b, Empty lookup table LUT
Output: Ensemble result $\{\tilde{\mathbf{Y}}_n\}$
Estimation Stage:
Obtain restoration predictions by $\mathbf{x}_{m,n} = \text{flatten}\left(f_m(\mathbf{X}_n)\right), \forall m \in \{1, ..., M\}$;
Append restoration predictions and ground-truths into $\mathbf{y}_{1:N}$ and $\mathbf{x}_{m,1:N}$ based on Eq. 2;
Define bin set space $\mathbb{B} = \{[0, b), [b, 2b), ..., [(T-1)b, 255]\}$;
for *each bin set* $(B_1, ...B_M) \in \mathbb{B}^M$ **do**
  Compute the partition map $\mathbf{R}_r = \prod_{m=1}^{M} \mathbf{I}_{B_m}(f_m(\mathbf{x}_n))$;
  Partition images and obtain range-wise patches $(\mathbf{y}_{r,1:N}, \mathbf{x}_{r,1,1:N}, ..., \mathbf{x}_{r,M,1:N})$ by Eq. 3;
  $(\alpha_{r,1}, ..., \alpha_{r,M}) \leftarrow \boxed{\text{MPEM}}(\mathbf{y}_{r,1:N}, \mathbf{x}_{r,1,1:N}, ..., \mathbf{x}_{r,M,1:N})$;
  Store $\text{LUT}[(B_1, ...B_M)] \leftarrow (\alpha_{r,1}, ..., \alpha_{r,M})$;
end
Inference Stage:
for *each test data* $\hat{\mathbf{X}}_n$ **do**
  for *each bin set* $(B_1, ...B_M) \in \mathbb{B}^M$ **do**
    Retrieve $(\alpha_{r,1}, ..., \alpha_{r,M}) \leftarrow \text{LUT}[(B_1, ...B_M)]$;
    Partition input as $\tilde{\mathbf{X}}_{r,m,n} \leftarrow \mathbf{R}_r \cdot f_m(\hat{\mathbf{X}}_n)$, where $\mathbf{R}_r = \prod_{m=1}^{M} \mathbf{I}_{B_m}(f_m(\hat{\mathbf{X}}_n))$;
    $\tilde{\mathbf{Y}}_{r,n} \leftarrow \sum_{m=1}^{M} \alpha_{r,m} \tilde{\mathbf{X}}_{r,m,n}$; /* Inner summation of Eq. 15 */
  end
  $\tilde{\mathbf{Y}}_n \leftarrow \sum_{r=1}^{T^M} \tilde{\mathbf{Y}}_{r,n}$; /* Outer summation of Eq. 15 */
end

A Simple Case with bin width=64



Result 2: Ablation Studies

Table 1: Ablation study of bin width b on Rain100H [74] with maximum step number 1000. "Runtime" is the average runtime [s].

b	16	32	64	96	128
Runtime	1.2460	0.1709	0.0265	0.0132	0.0059
PSNR	31.745	31.739	31.720	31.713	31.725
SSIM	0.9093	0.9095	0.9094	0.9093	0.9093

Table 2: Ablation study of maximum step number in the EM algorithm on Rain100H [74] with $b = 32$. "Time" is the time of EM algorithm [s].

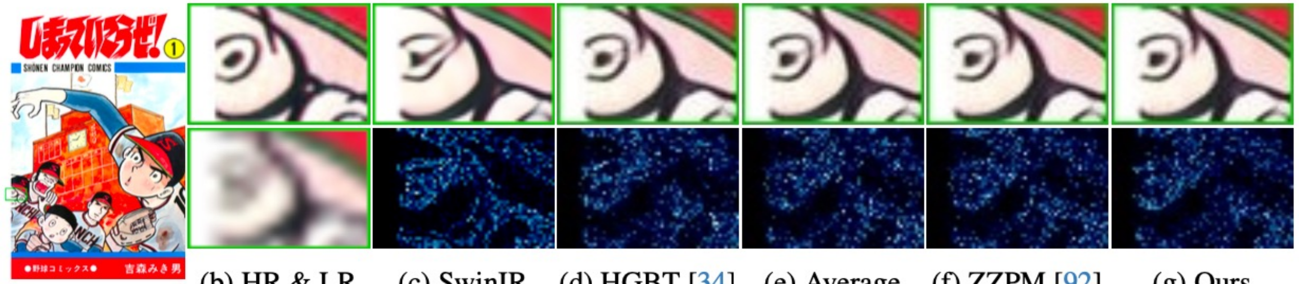
#step	10	100	500	1000	10000
Time	12.108	28.516	30.409	30.518	30.524
PSNR	31.734	31.738	31.738	31.739	31.739
SSIM	0.9093	0.9094	0.9095	0.9095	0.9095

Table 6: The average runtime per image in seconds of the ensemble methods on Rain100H [74].

Method	Bagging [6]	AdaBoost [22]	RForest [7]	GBDT [23]	HGBT [34]	Average	ZZPM [92]	Ours
Runtime	1.0070	1.1827	9.8598	1.2781	0.1773	0.0003	0.0021	0.1709

Quantitative Comparison: Super-Resolution

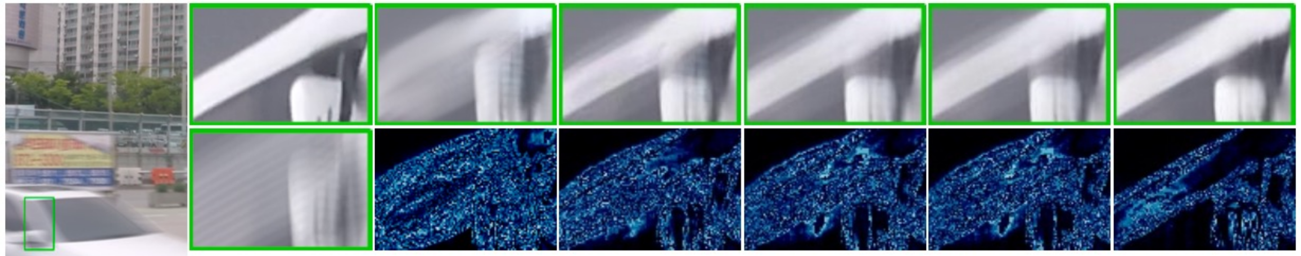
	Datasets	Set5 [5]		Set14 [83]		BSDS100 [47]		Urban100 [30]		Manga109 [48]	
	Metrics	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
Base	SwinIR [39]	32.916	0.9044	29.087	0.7950	27.919	0.7487	27.453	0.8254	32.024	0.9260
	SRFormer [93]	32.922	0.9043	29.090	0.7942	27.914	0.7489	27.535	0.8261	32.203	0.9271
	MambaIR [27]	33.045	0.9051	29.159	0.7958	27.967	0.7510	27.775	0.8321	32.308	0.9283
Regr.	Bagging [6]	33.006	0.9050	29.119	0.7950	27.946	0.7498	27.546	0.8273	32.154	0.9270
	AdaBoost [22]	33.072	0.9049	29.175	0.7959	27.975	0.7503	27.786	0.8302	32.457	0.9286
	RForest [7]	33.032	0.9052	29.158	0.7954	27.964	0.7500	27.640	0.8287	32.287	0.9279
	GBDT [23]	33.085	0.9050	29.196	0.7956	27.980	0.7500	27.792	0.8311	32.467	0.9285
	HGBT [34]	33.078	0.9051	29.201	0.7959	27.984	0.7502	27.783	0.8310	32.444	0.9282
IR.	Average	33.097	0.9057	29.202	**0.7964**	27.983	0.7506	27.785	0.8313	32.466	0.9290
	RefESR [31]	33.091	0.9052	29.172	0.7960	27.972	0.7504	27.785	0.8312	32.447	0.9288
	ZZPM [92]	33.094	0.9057	29.203	0.7963	27.981	0.7506	27.786	0.8313	32.467	0.9290
	EnsIR (Ours)	**33.103**	**0.9058**	**29.205**	**0.7964**	**27.984**	**0.7507**	**27.795**	**0.8315**	**32.468**	**0.9291**



(a) Image | (b) HR & LR PSNR/SSIM | (c) SwinIR 33.174/0.9674 | (d) HGBT [34] 33.919/0.9683 | (e) Average 33.873/0.9695 | (f) ZZPM [92] 33.874/0.9694 | (g) Ours 33.935/0.9696

Quantitative Comparison: Deblurring

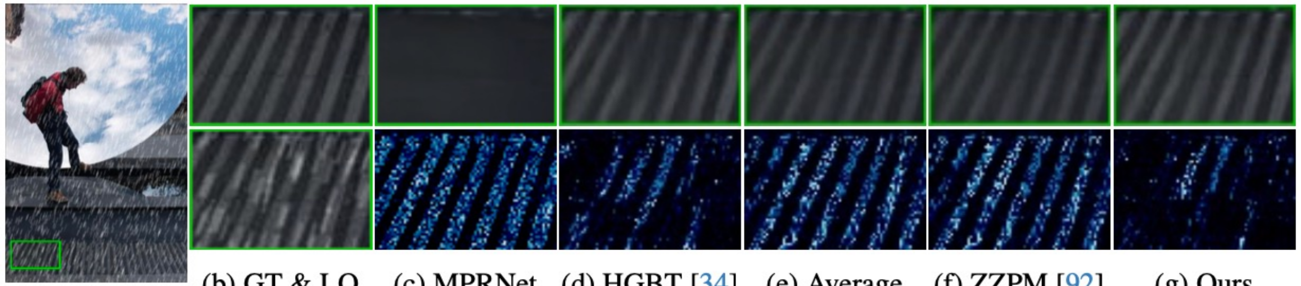
	Datasets	GoPro [51]		HIDE [60]		RealBlur-R [58]		RealBlur-J [58]	
	Metrics	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
Base	MPRNet [82]	32.658	0.9362	30.962	0.9188	33.914	0.9425	26.515	0.8240
	Restormer [81]	32.918	0.9398	31.221	0.9226	33.984	0.9463	26.626	0.8274
	DGUNet [50]	33.173	0.9423	31.404	0.9257	33.990	0.9430	26.583	0.8261
Regr.	Bagging [6]	33.194	0.9418	31.437	0.9250	34.033	0.9456	26.641	0.8277
	AdaBoost [22]	33.205	0.9412	31.449	0.9251	34.035	0.9455	26.652	0.8280
	RForest [7]	33.173	0.9416	31.439	0.9247	34.039	0.9457	26.647	0.8280
	GBDT [23]	33.311	0.9418	31.568	0.9256	34.052	0.9465	26.684	0.8285
	HGBT [34]	33.323	0.9427	31.583	0.9267	33.986	0.9436	26.684	0.8296
IR.	Average	33.330	0.9436	31.579	0.9277	34.090	0.9471	26.689	**0.8309**
	ZZPM [92]	33.332	0.9436	31.580	0.9277	34.057	0.9468	26.688	0.8308
	EnsIR (Ours)	**33.345**	**0.9438**	**31.590**	**0.9278**	**34.089**	**0.9472**	**26.690**	**0.8309**



(a) Image
(b) GT & LQ PSNR/SSIM
(c) Restormer 29.980/0.9377
(d) HGBT [34] 32.110/0.9481
(e) Average 32.177/0.9479
(f) ZZPM [92] 32.170/0.9475
(g) Ours 32.584/0.9486

Quantitative Comparison: Deraining

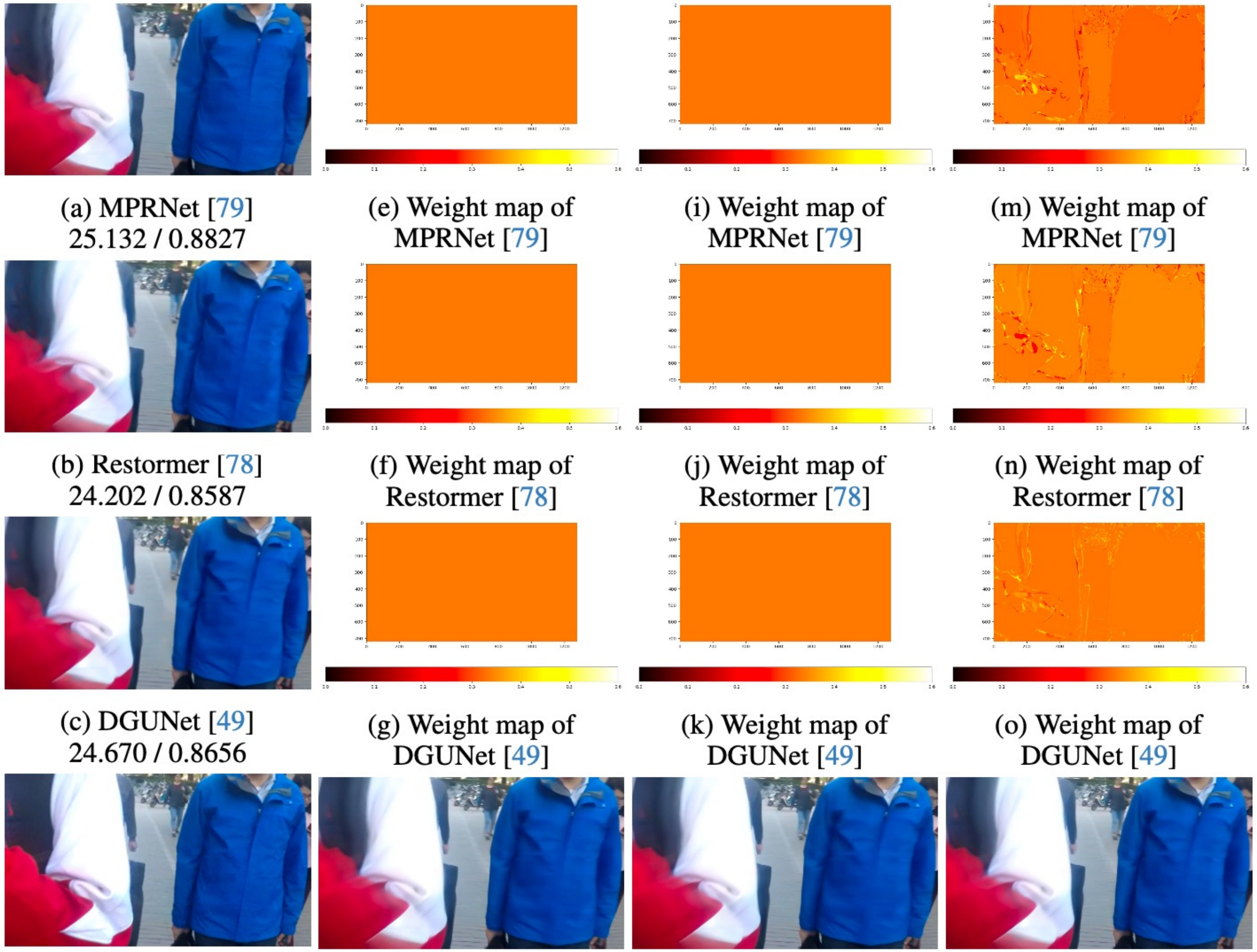
	Datasets	Rain100H [74]		Rain100L [74]		Test100 [88]		Test1200 [86]		Test2800 [25]	
	Metrics	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
Base	MPRNet [82]	30.428	0.8905	36.463	0.9657	30.292	0.8983	32.944	0.9175	33.667	0.9389
	MAXIM [67]	30.838	0.9043	38.152	0.9782	31.194	0.9239	32.401	0.9240	33.837	0.9438
	Restormer [81]	31.477	0.9054	39.080	0.9785	32.025	0.9237	33.219	0.9270	34.211	0.9449
Regr.	Bagging [6]	31.461	0.9001	39.060	0.9782	31.865	0.9107	33.115	0.9152	34.216	0.9446
	AdaBoost [22]	31.472	0.9006	39.067	0.9782	31.866	0.9112	33.117	0.9153	34.221	0.9443
	RForest [7]	31.492	0.9012	39.089	0.9784	31.900	0.9127	33.147	0.9169	34.224	0.9447
	GBDT [23]	31.581	0.9058	39.044	0.9778	32.001	0.9236	33.276	0.9274	34.211	0.9446
	HGBT [34]	31.698	0.9075	39.115	0.9784	31.988	0.9241	33.305	0.9282	34.229	0.9450
IR.	Average	31.681	0.9091	38.675	0.9770	31.626	0.9225	33.427	0.9286	34.214	0.9449
	ZZPM [92]	31.689	0.9091	38.725	0.9771	31.642	0.9227	33.434	0.9286	34.231	0.9450
	EnsIR (Ours)	**31.739**	**0.9095**	**39.216**	**0.9792**	**32.064**	**0.9258**	**33.445**	**0.9289**	**34.245**	**0.9451**



(a) Image (b) GT & LQ PSNR/SSIM (c) MPRNet 34.008/0.9359 (d) HGBT [34] 37.052/0.9615 (e) Average 36.775/0.9603 (f) ZZPM [92] 36.777/0.9601 (g) Ours 37.372/0.9639

Visualization

- Weight Map



(a) MPRNet [79]
25.132 / 0.8827

(e) Weight map of MPRNet [79]

(i) Weight map of MPRNet [79]

(m) Weight map of MPRNet [79]

(b) Restormer [78]
24.202 / 0.8587

(f) Weight map of Restormer [78]

(j) Weight map of Restormer [78]

(n) Weight map of Restormer [78]

(c) DGUNet [49]
24.670 / 0.8656

(g) Weight map of DGUNet [49]

(k) Weight map of DGUNet [49]

(o) Weight map of DGUNet [49]

(d) Ground-truth
PSNR / SSIM

(h) Averaging
25.232 / 0.8817

(l) ZZPM [89]
25.233 / 0.8837

(p) Ours w/ $b = 32$
25.247 / 0.8840

Figure 4: A sample of weight visualizations on HIDE [59]. Base models are DGUNet [49], MPR-Net [79], and Restormer [78]. The first column shows the base model predictions and ground-truth. The second column shows the ensemble weights and result of the averaging strategy. The third column shows the ensemble weights and result of ZZPM [89]. The last column shows the ensemble weights and result of our method.

Visualization

- Feature by ResNet110 and PCA

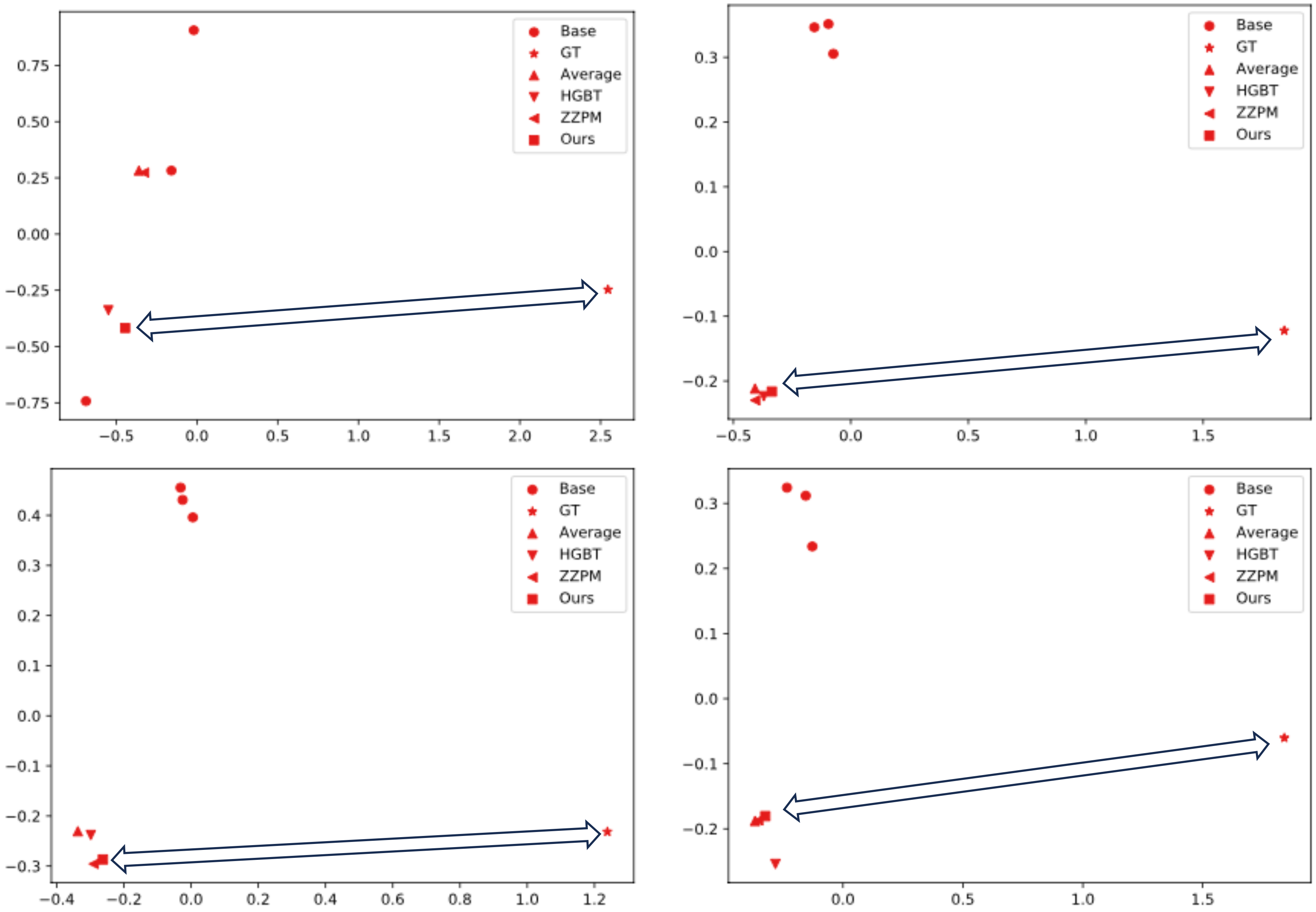
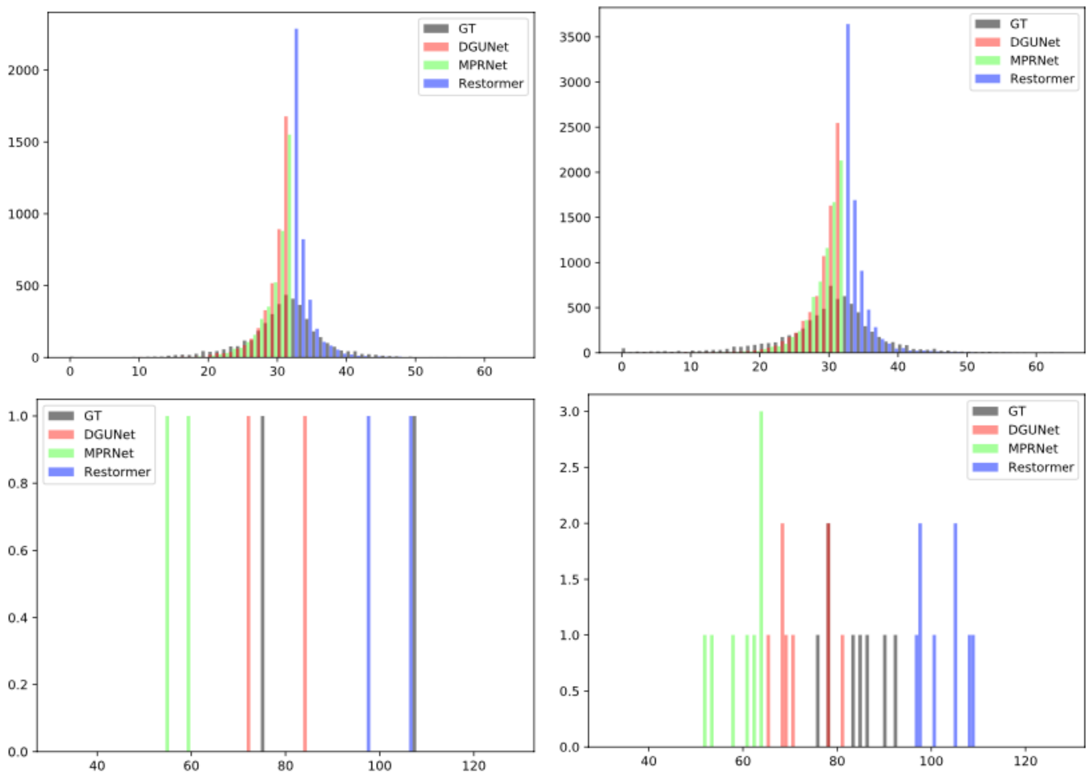
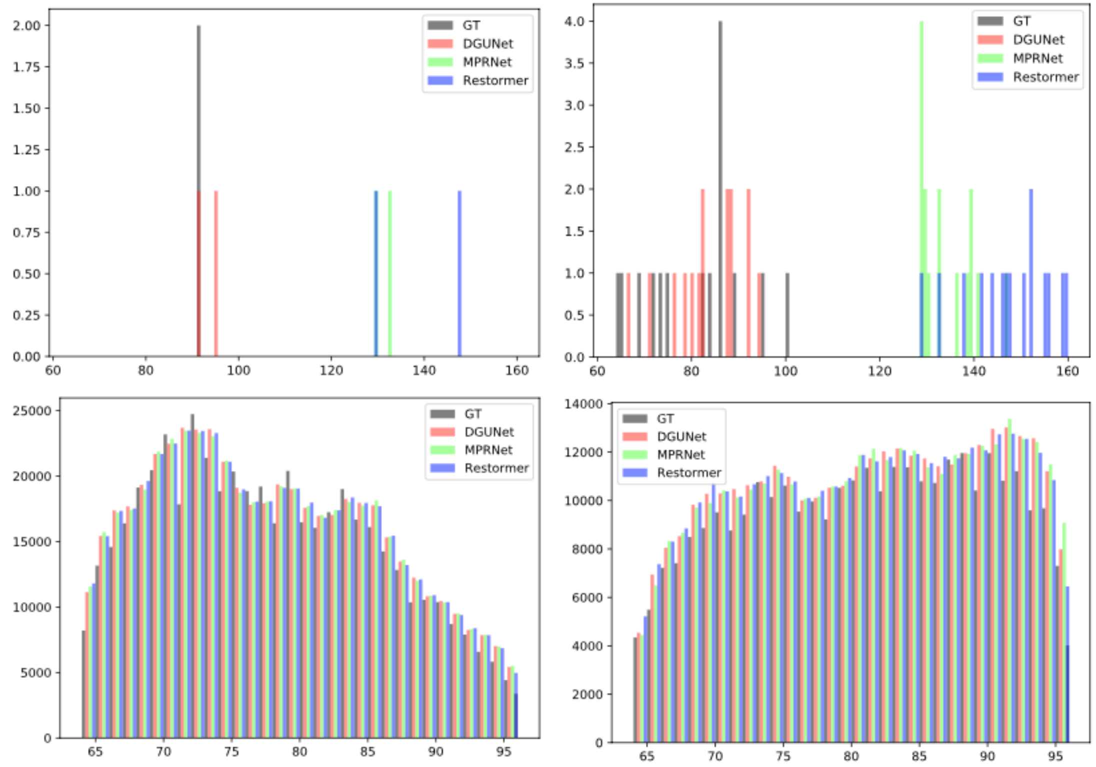


Figure 5: A sampled group of feature visualizations on HIDE [59]. “Base” denotes the features of three base models, i.e., DGUNet [49], MPRNet [79], and Restormer [78].

Visualization

- Pixel Distribution within bin set





(a) Reference set

(b) Test set

Figure 6: A group of distribution visualizations on HIDE [59]. The bin sets of the first row is $(B_1 = [0, 32), B_2 = [0, 32), B_3 = [32, 64))$. The bin sets of the second row is $(B_1 = [64, 96), B_2 = [32, 64), B_3 = [96, 128))$. The bin sets of the third row is $(B_1 = [64, 96), B_2 = [128, 160), B_3 = [128, 160))$. The bin sets of the last row is $(B_1 = [64, 96), B_2 = [64, 96), B_3 = [64, 96))$. Base models are DGUNet [49], MPRNet [79], and Restormer [78].

Thank You