Supplementary Material for
Exploiting Diffusion Prior for Task-driven Image Restoration

Jaeha Kim[1] Junghun Oh[1] Kyoung Mu Lee[1,2]
[1]Dept. of ECE&ASRI, [2]IPAI, Seoul National University, Korea
jhkim97s2@gmail.com, {dh6dh, kyoungmu}@snu.ac.kr

In this supplementary document, we show the additional details, results, and ablation studies omitted from the main manuscript due to the lack of space:

- S1. Degradation details
- S2. Training details
- S3. Impact of pre-restoration network
- S4. Training algorithm
- S5. Benefit of using two feature spaces in HLF
- S6. Computational cost of the EDTR
- S7. Output stochasticity of the EDTR
- S8. Comparison with DiffBIR for detection
- S9. SR4IR combined with SD
- S10. Details for the previous works
- S11. Additional ablation studies
- S12. Further visualization results

S1. Degradation details

We provide the detailed image degradation settings used in our main experiments (Table 1 of our main manuscript).

- **Mixture-*A*:** Bilinear downsampling with a scale factor of $\times 8$ is applied, followed by JPEG compression with a quality factor of 75.
- **Mixture-*B*:** The Gaussian blur, bilinear downsampling, Gaussian noise, and JPEG compression are sequentially applied. Gaussian blur kernel size, downsampling ratio, Gaussian noise standard deviation, and JPEG quality factor are randomly selected within the ranges $[0.1, 8.0]$, $[1, 16]$, $[0, 10]$, and $[50, 100]$, respectively.

Note that after applying degradation, all degraded images are resized back to their original resolution using bilinear interpolation to match the size of high-quality images, following the CodeFormer [68].

S2. Training details

The EDTR and high-level vision task networks are optimized over 10k iterations using the AdamW [69] and SGD optimizers, respectively. The learning rates are set to 10^{-4} for EDTR and 5×10^{-3} for the task network, with a cosine annealing [70] schedule. For each degraded dataset, the pre-restoration network (*i.e.*, SwinIR [27]) is trained with the same number of iterations using the AdamW optimizer and a learning rate of 10^{-4}. The training batch size is 32 for image classification and 16 for other tasks. The image resolution is fixed at 512×512 for all tasks. If an image is smaller than 512×512, we pad it to meet the resolution and then crop it back to its original size. We calculate the HLF loss using the output of the feature extractor for each task network, *e.g.*, for image classification, the feature directly preceding the final fully connected layer in ResNet [15]. When computing the training loss for the task network in Equation (10), the balancing hyper-parameter α is set to 1 for classification, 0.5 for segmentation, and 0.2 for detection. The task loss is defined as cross-entropy loss for classification and segmentation, and as a combination of classification, Region-of-Interest regression, objectness, and Region Proposal Network box regression losses for detection.

S3. Impact of pre-restoration network

Table S1 presents the performance of EDTR-1 step along with various pre-restoration (*i.e.*, pixel-error) networks. We observe that a more powerful pre-restoration network, which achieves a higher pre-restoration PSNR, leads to better high-level vision task performance. These results support our claim that removing degradation artifacts as much as possible before applying the diffusion prior is crucial for effectively harnessing the power of the diffusion prior in the TDIR, as discussed in Section 3.2. In addition, this result also indicates that developing a classical IR model aimed at achieving higher PSNR is also relevant to the TDIR.

Pre-restoration network	EDSR [28]	RRDBNet [53]	SwinIR (**Used**)
Acc$_\uparrow$ / Pre-restoration PSNR$_\uparrow$	66.8 / 24.45	67.3 / 24.59	68.8 / 25.04

Table S1. **Impact of pre-restoration network in classification.**

S4. Training algorithm

Algorithm S1 presents the detailed training procedure for jointly training EDTR and task network $\mathcal{H}$, which are introduced in Section 3.2 and 3.3. Note that EDTR and task network are trained alternately, *i.e.*, in one training iteration, EDTR is trained first, followed by an update to the $\mathcal{H}$.

Algorithm S1 Training algorithm for jointly training EDTR and the task network $\mathcal{H}$

Input: Trainable parameter θ_{EDTR} for EDTR, $\theta_{\mathcal{H}}$ for task network $\mathcal{H}$, pre-restoration network $\mathcal{R}_{\text{pix}}$, denoising network combined with ControlNet ϵ_θ, VAE encoder $\mathcal{E}$ and decoder $\mathcal{D}$, set of HQ patches $\mathcal{I}_{\text{HQ}}$, image degradation model Deg, timestep for partial diffusion t_p, number of denoising steps n, variance schedule of Gaussian noise $\beta_t \in (0, 1)$, high- and low-frequency wavelet components $\mathbf{H}$ and $\mathbf{L}$, total training iterations N, learning rates η_{EDTR} and $\eta_{\mathcal{H}}$

$q(z_t|z_{t-1}) \coloneqq \mathcal{N}(\sqrt{1-\beta_t}\, z_{t-1}, \beta_t\, \mathbf{I})$, $\alpha_t = 1 - \beta_t$, $\bar{\alpha}_t = \prod_{i=1}^{t} \alpha_i$, $\epsilon \sim \mathcal{N}(0, \mathbf{I})$
$\mathcal{T} = [t_p, \lfloor \frac{t_p \cdot (n-1)}{n} \rfloor, ..., \lfloor \frac{t_p}{n} \rfloor]$ // Used n timesteps for EDTR
for $i = 1 : N$ **do**
  $I_{\text{HQ}} \sim \mathcal{I}_{\text{HQ}}$, $I_{\text{LQ}} = Deg(I_{\text{HQ}})$ // Sample HQ images and generate LQ images
  $z_{\text{pre-res}} = \mathcal{E}(\mathcal{R}_{\text{pix}}(I_{\text{LQ}}))$ // Pre-restoration and encoding
  `# Training EDTR`
  $t \sim \text{Uniform}(\mathcal{T})$ // Sample timestep t
  $z_{t,\text{partial}} = q(z_t|z_0 = z_{\text{pre-res}})$ // Partial diffusion with timestep t, Equation (4)
  $z_{\text{diff-res}} = (z_{t,\text{partial}} - \sqrt{1-\bar{\alpha}_t}\epsilon_\theta(z_{t,\text{partial}}, t, z_{\text{pre-res}}))/\sqrt{\bar{\alpha}_t}$ // One-step denoising
  $I_{\text{EDTR,train}} = \mathbf{H}(\mathcal{D}(z_{\text{diff-res}})) + \mathbf{L}(\mathcal{R}_{\text{pix}}(I_{\text{LQ}}))$ // RGB image decoding with color correction, Equation (6)
  $\theta_{\text{EDTR}} \leftarrow \theta_{\text{EDTR}} - \eta_{\text{EDTR}} \nabla_{\theta_{\text{EDTR}}} \mathcal{L}_{\text{HLF}}$ // Update θ_{EDTR} using HLF loss, Equation (7)
  `# Training task network`
  $z_{t_p,\text{partial}} = q(z_{t_p}|z_0 = z_{\text{pre-res}})$ // Partial diffusion with timestep t_p, Equation (4)
  `# n-step denoising, we set n to a small value for short-step (e.g., 1, 4)`
  for $j = 0 : (n-1)$ **do**
    $z_{\text{diff-res}} = (z_{\mathcal{T}[j],\text{partial}} - \sqrt{1-\bar{\alpha}_{\mathcal{T}[j]}}\epsilon_\theta(z_{\mathcal{T}[j],\text{partial}}, \mathcal{T}[j], z_{\text{pre-res}}))/\sqrt{\bar{\alpha}_{\mathcal{T}[j]}}$ // One-step denoising
    if $j \neq (n-1)$ **then**
      $z_{\mathcal{T}[j+1],\text{partial}} = q(z_{\mathcal{T}[j+1]}|z_{\mathcal{T}[j]}, z_0 = z_{\text{diff-res}})$ // Adding noise
    end if
  end for
  $I_{\text{EDTR}} = \mathbf{H}(\mathcal{D}(z_{\text{diff-res}})) + \mathbf{L}(\mathcal{R}_{\text{pix}}(I_{\text{LQ}}))$ // RGB image decoding with color correction, Equation (6)
  $\theta_{\mathcal{H}} \leftarrow \theta_{\mathcal{H}} - \eta_{\mathcal{H}} \nabla_{\theta_{\mathcal{H}}} (\mathcal{L}_{\text{task}} + \alpha\, \mathcal{L}_{\text{FM}})$ // Update $\theta_{\mathcal{H}}$ using task loss and FM loss, Equations (8) and (9)
end for

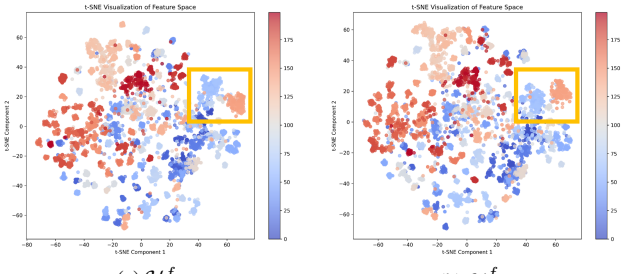
S5. Benefit of using two feature spaces in HLF

We propose the HLF loss, which calculates the distance in the feature space of two task networks, effectively guiding the diffusion prior to restore task-relevant details. To evaluate the effectiveness of using the feature space from both task networks, we compare the performance of EDTR trained with the loss for each single task network. Specifically, we compare two cases: using only $\mathcal{H}^f$ and using only $\mathcal{H}^f_{\text{HQ}}$, in the original HLF loss (Equation (7)). Note that using only $\mathcal{H}^f$ is the same configuration (*i.e.*, TDP loss) proposed in the previous TDIR method SR4IR [20].

Table S2 presents the performance of EDTR under different training loss settings. The EDTR model trained with our HLF loss achieves the best performance in both task accuracy and visual quality. We claim that this is because the HLF loss extracts complementary information from both task networks, leading to improved performance compared to using a single task network. Figure S1 further validates our claim by visualizing the feature spaces of two task networks, $\mathcal{H}^f$ and $\mathcal{H}^f_{\text{HQ}}$. Specifically, we use the t-SNE [71] method for feature space visualization. Although the two feature spaces exhibit notable similarity due to the feature matching loss, apparent differences exist between the two feature spaces. For example, as highlighted in the yellow box in Figure S1, the relationship between the red and blue

Training loss	$\mathcal{H}^f$	$\mathcal{H}^f_{\text{HQ}}$	Acc$_\uparrow$ (%)	Q-Align$_\uparrow$
Only $\mathcal{H}^f$	✓	✗	67.8	3.36
Only $\mathcal{H}^f_{\text{HQ}}$	✗	✓	68.0	3.26
HLF **(Ours)**	✓	✓	**68.8**	**3.48**

Table S2. **EDTR performance on classification with different training losses.** The EDTR-1 step model is used.



(a) $\mathcal{H}^f$ (b) $\mathcal{H}^f_{\text{HQ}}$

Figure S1. **t-SNE visualizations of the feature space.** The feature spaces are from ResNet for classification. Each color of the point represents a classification label.

point groups differs in $\mathcal{H}^f$ and $\mathcal{H}^f_{\text{HQ}}$. This suggests that incorporating two feature spaces in HLF loss can provide complementary information for the TDIR, leading to improved performance.

S6. Computational cost of the EDTR

Table S3 presents the computational cost of our EDTR. The throughput and VRAM usage are measured on 512×512 resolution images using a single NVIDIA RTX A6000 GPU and an Intel Xeon Gold 6226R CPU. Notably, despite incorporating the large SD, EDTR achieves reasonable throughputs of 3.79 and 2.23 img/s for 1-step and 4-step settings, respectively, owing to its efficient short-step denoising.

IR methods	Throughput (img/s)	# of parameters (B)	VRAM usage (MB)
DiffBIR [30] (50 step)	0.31	1.683 (0.363 + 1.320)	9310
EDTR-1 step **(Ours)**	3.79	1.683 (0.413 + 1.270)	9310
EDTR-4 step **(Ours)**	2.23	1.683 (0.413 + 1.270)	9310

Table S3. **Computational cost of the EDTR.** The gray number represents the frozen parameters in SwinIR and SD. EDTR has more trainable parameters due to its trainable VAE decoder.

S7. Output stochasticity of the EDTR

The outputs of EDTR exhibit stochasticity during inference due to the random noise ϵ in the partial diffusion process, as described in Equation (4). Table S4 presents the average and standard deviation of the EDTR output metrics. While the performance of high-level vision tasks shows some variability, image quality metrics remain highly consistent, with a standard deviation of less than 0.002. Therefore, we report image quality metrics from a single inference while averaging the results of four inferences to ensure reliability when evaluating the task performance.

Methods	Acc$_\uparrow$ (%)	NIQE$_\downarrow$	Q-Align$_\uparrow$	PSNR$_\uparrow$
EDTR-1 step **(Ours)**	68.8 / 0.249	4.75/ 0.002	3.48/ 0.002	23.03 / 0.001

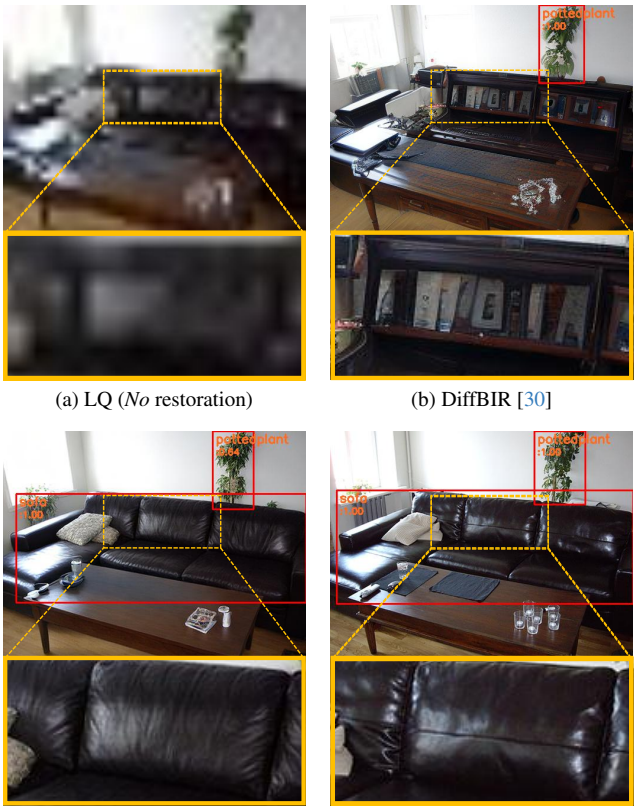
Table S4. **Output stochasticity of EDTR-1 step.** We report the average / standard deviation over 10 inference runs for each metric on the image classification task under Mixture-*B* degradation.

S8. Comparison with DiffBIR for detection

Table S5 compares EDTR with the conventional SD-based IR method (*i.e.*, DiffBIR [30], Exp-(1) setting in Table 2 of our main manuscript) for object detection. Figure S2 illustrates that, although the conventional SD-based IR method restores the image in a visually pleasing manner, its generative prior can produce undesirable patterns, leading to misdetection results. For example, the area highlighted in the yellow box in Figure S2 is part of a sofa, but the conventional SD-based IR method restores it to resemble a bookshelf, resulting in a failure to detect the sofa. These results further validate that the *diffusion prior must be carefully managed* to effectively restore task-relevant details.

Methods	mAP$_\uparrow$ (%)	Q-Align$_\uparrow$
Conventional SD-based IR method [30]	25.9	3.87
EDTR-1 step **(Ours)**	30.6	3.64
EDTR-4 step **(Ours)**	**31.9**	**4.02**

Table S5. **Performance comparison for detection (Mixture-*B*).**



(a) LQ (*No* restoration) (b) DiffBIR [30]

(c) **EDTR (Ours)** (d) HQ (Oracle)

Figure S2. **Comparison between EDTR-4 step and conventional SD-based IR method on object detection.**

S9. SR4IR combined with SD

Table S6 presents the results of directly incorporating SD into the previous state-of-the-art TDIR method, SR4IR [20], with the results visualized in Figure 1c. Note that this approach differs from Exp-(2) in Table 2, including the use of FM loss and other techniques (*e.g.*, TDP and CQMix, which were introduced in SR4IR). Despite the incorporation of a strong diffusion prior, it performs even worse than the original SR4IR. These results further highlight that even with a strong diffusion prior, effectively handling it to restore task-relevant details is a crucial challenge.

Methods	Acc$_\uparrow$ (%)	NIQE$_\downarrow$	Q-Align$_\uparrow$	PSNR$_\uparrow$
SR4IR [20]	63.4	6.08	3.11	23.62
SR4IR [20] + SD [41]	57.6	6.01	2.35	16.92
EDTR-1 step **(Ours)**	68.8	4.75	3.48	23.03

Table S6. **Performance of the SR4IR combined with SD.**

S10. Details for the previous works

TDSR. Since the official code for TDSR [14] is not available, we re-implemented it ourselves. We use TDSR-0.01, which employs a pixel loss ratio of 1.0 and a task loss ratio of 0.01. Unlike the original TDSR, which is limited to object detection, we extend it to image classification and semantic segmentation, utilizing the respective high-level vision task losses, *e.g.*, cross-entropy loss for image classification. We adopt SwinIR [27] as the restoration model.

RSRSSN. Since the official code for RSRSSN [67] is not available, we re-implemented it ourselves. Following the paper, we train the restoration model and task network in an end-to-end manner using feature map multi-box loss and task loss, with respective weight ratios of 0.1 and 1.0. As with our re-implementation of TDSR, we extend RSRSSN to include image classification and semantic segmentation. We additionally incorporate pixel loss during training, as we observed significantly degraded image quality without it. We use SwinIR as the restoration model.

SR4IR. We use the official code from SR4IR [20] to obtain the results. SwinIR is used as the restoration model.

S11. Additional ablation studies

Exp	Methods	Acc$_\uparrow$ (%)	NIQE$_\downarrow$	Q-Align$_\uparrow$	PSNR$_\uparrow$
(1)	**EDTR (Ours)**	68.8	4.75	3.48	23.03
(2)	*without* trainable $\mathcal{D}$	67.7	6.03	3.55	23.61
(3)	*without* color correction	68.4	4.89	3.56	22.84

Table S7. **Performance of EDTR-1 step under various settings.**

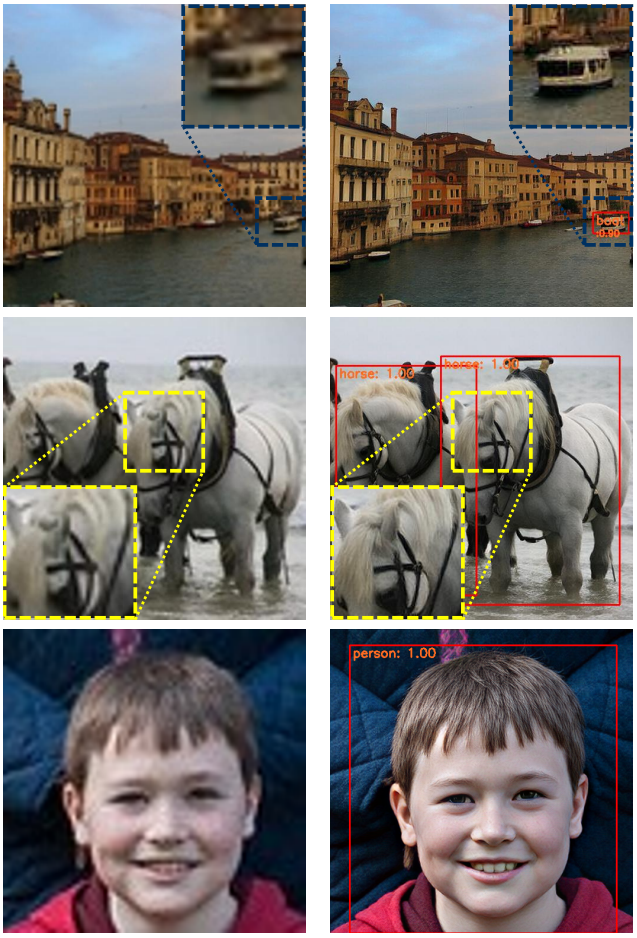
Trainable decoder. The Exp-(2) result in Table S7 shows the EDTR performance without a trainable VAE decoder. Although EDTR without a trainable decoder obtains a higher PSNR amount of 0.58 dB, it scores 1.1% lower in classification accuracy and shows a worse NIQE score; therefore, we opt for the VAE decoder to be trainable.

Wavelet color correction. The Exp-(3) result in Table S7 shows the EDTR performance without Wavelet color correction, as introduced in Equation (6) of the main manuscript. EDTR without Wavelet color correction achieves a slightly lower classification accuracy (-0.4%) and a PSNR drop of 0.19 dB. Therefore, we opt to include Wavelet color correction in our EDTR.

S12. Further visualization results

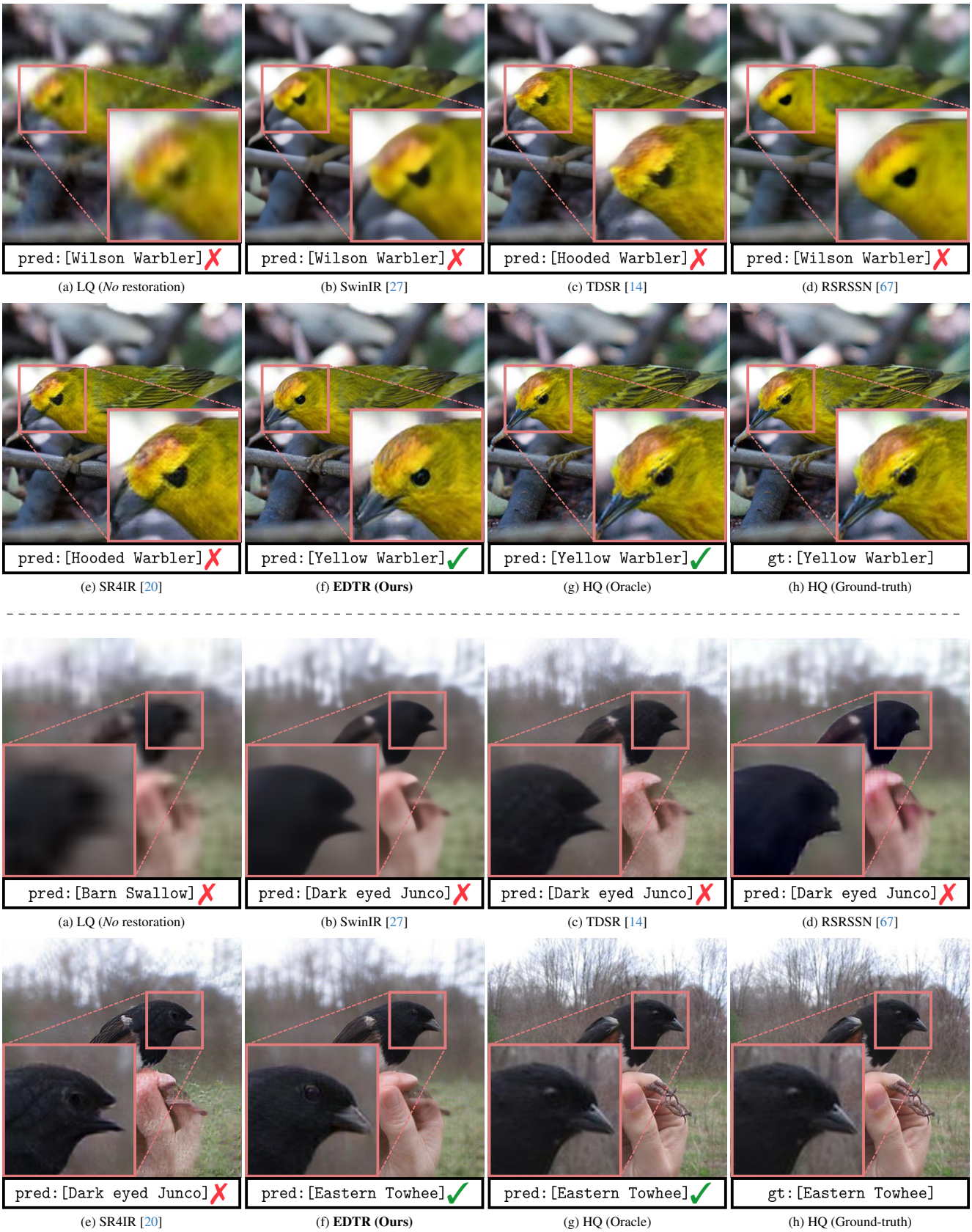
Benchmark datasets. Figures S4, S5, and S6 provide additional visualizations across various high-level vision tasks. We include further comparisons with SwinIR [27], TDSR [14], RSRSSN [67], and ground-truths, expanding on Figure 5 from our main manuscript. Figure S4 shows that EDTR successfully restores fine details of the bird, resulting in the only correct classification. Figure S5 demonstrates that EDTR restores the horse's mane and the bottle's shape, producing the most accurate segmentation results. Figure S6 illustrates that EDTR restores the horse's eye and the bottle's shape, achieving the only correct detection. These results clearly demonstrate that EDTR, which effectively leverages the powerful diffusion prior, is highly valuable for addressing the TDIR problem in challenging degraded scenarios.

Real-world images. Figures S3 provide additional visualizations of real-world object detection. Our EDTR successfully restores the shape of the boat, the horse's face, and the boy's face using the powerful diffusion prior, enabling successful detection in real-world degraded images. These results demonstrate the strong generalizability and practicality of our method.



(a) LQ (*No* restoration) (b) **EDTR (Ours)**

Figure S3. **Additional real-world object detection results and visualization of the restored image.** The first, second, and third images are from `"14.png"`, `"01.png"` and `"21.png"` in the real-world image set RealPhoto60 [61]. The EDTR-4 step model is used for visualization.



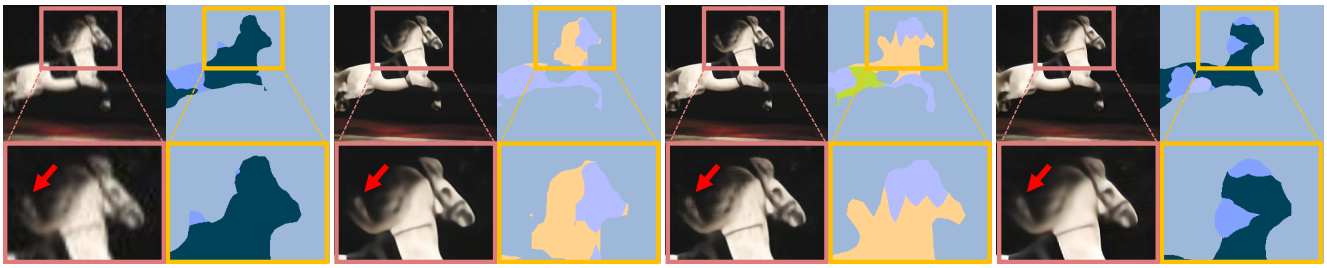
(a) LQ (*No* restoration) (b) SwinIR [27] (c) TDSR [14] (d) RSRSSN [67]

(e) SR4IR [20] (f) **EDTR (Ours)** (g) HQ (Oracle) (h) HQ (Ground-truth)

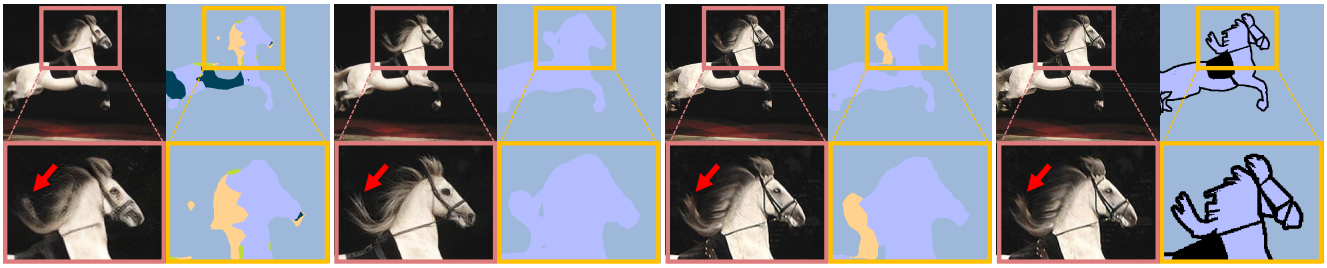
(a) LQ (*No* restoration) (b) SwinIR [27] (c) TDSR [14] (d) RSRSSN [67]

(e) SR4IR [20] (f) **EDTR (Ours)** (g) HQ (Oracle) (h) HQ (Ground-truth)

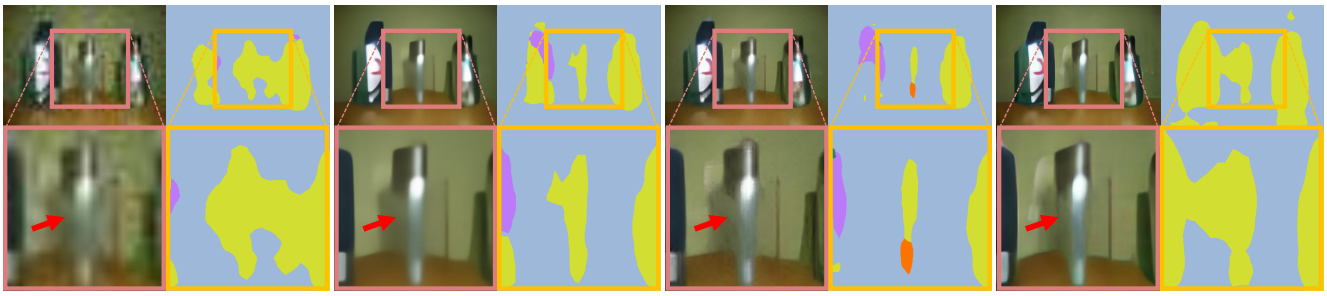
Figure S4. **Further visualization of images and image classification results on degraded LQ (Mixture-*B*) images.** We show the restored images and the corresponding predicted or ground-truth labels. The EDTR-1 step model is used for visualization.



(a) LQ (*No* restoration) (b) SwinIR [27] (c) TDSR [14] (d) RSRSSN [67]



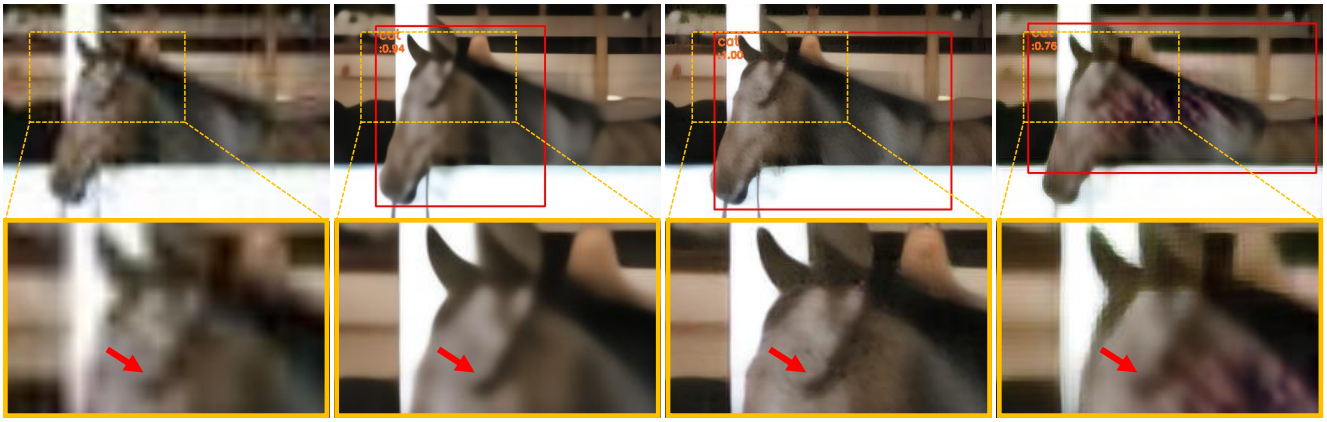
(e) SR4IR [20] (f) **EDTR (Ours)** (g) HQ (Oracle) (h) HQ (Ground-truth)



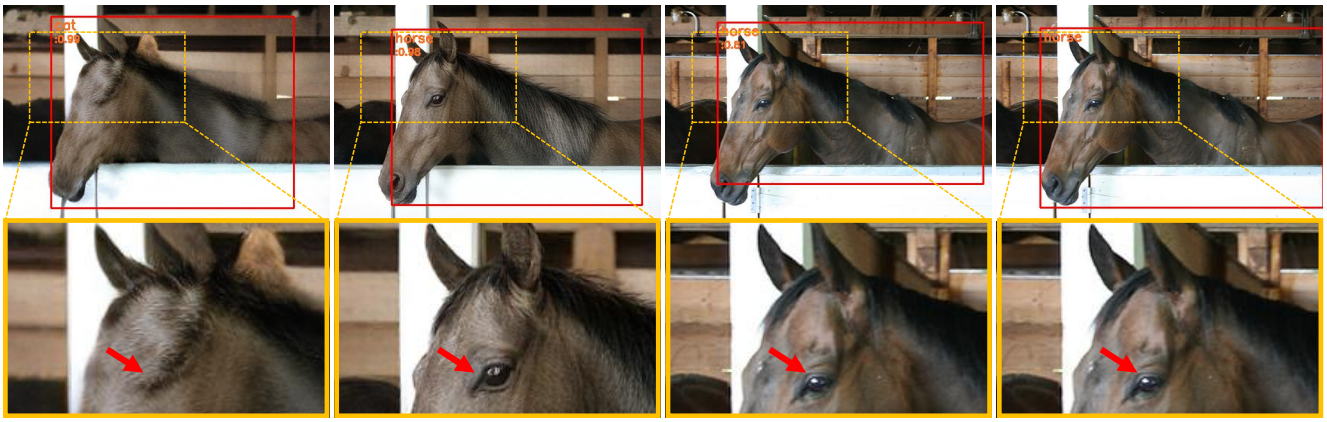
(a) LQ (*No* restoration) (b) SwinIR [27] (c) TDSR [14] (d) RSRSSN [67]

(e) SR4IR [20] (f) **EDTR (Ours)** (g) HQ (Oracle) (h) HQ (Ground-truth)

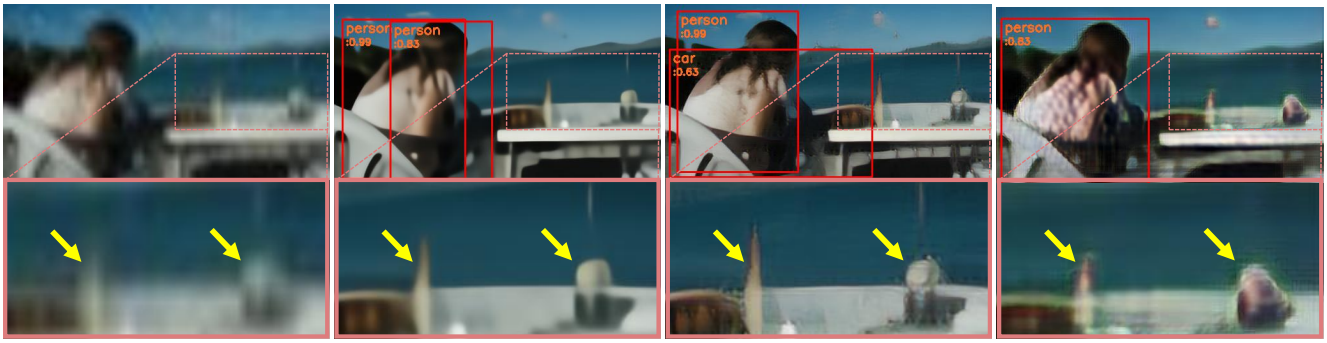
Figure S5. **Further visualization of images and semantic segmentation results on degraded LQ (Mixture-*B*) images.** We show the restored images and the corresponding predicted or ground-truth labels. The black line in the ground-truth segmentation map indicates "don't care" regions. The EDTR-4 step model is used for visualization.



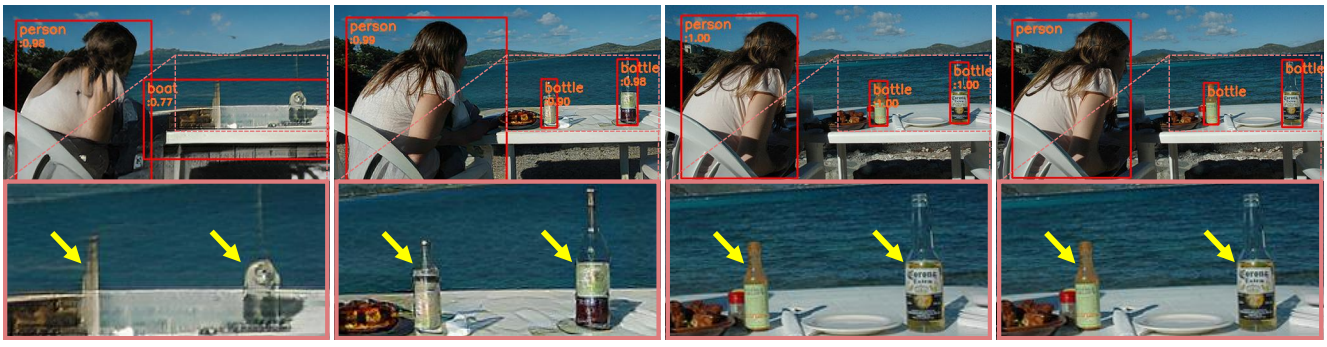
(a) LQ (*No* restoration) (b) SwinIR [27] (c) TDSR [14] (d) RSRSSN [67]



(e) SR4IR [20] (f) **EDTR (Ours)** (g) HQ (Oracle) (h) HQ (Ground-truth)



(a) LQ (*No* restoration) (b) SwinIR [27] (c) TDSR [14] (d) RSRSSN [67]



(e) SR4IR [20] (f) **EDTR (Ours)** (g) HQ (Oracle) (h) HQ (Ground-truth)

Figure S6. **Further visualization of images and object detection results on degraded LQ (Mixture-*B*) images.** We show the restored images and the corresponding predicted or ground-truth labels. The EDTR-4 step model is used for visualization.

References

[69] Loshchilov Ilya and Hutter Frank. Decoupled weight decay regularization. In *ICLR*, 2019.

[70] Loshchilov I. and Hutter F. Sgdr: Stochastic gradient descent with warm restarts. In *ICLR*, 2017.

[71] Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. In *JMLR*, 2008.