NanoSD: Edge Efficient Foundation Model for Real Time Image Restoration

Supplementary Material

1. Analysis of Generative Prior in NanoSD

A central claim of this work is that NanoSD retains the generative prior inherited from the original SD 1.5 backbone, even after undergoing distillation and downstream task adaptation. To validate this claim, we perform comprehensive qualitative and quantitative analyses, comparing NanoSD with both SD 1.5 and a computationally equivalent UNet-based restoration baseline.

1.1. Exploring the Latent Space Manifold

To evaluate whether NanoSD preserves the structured latent manifold inherent to SD 1.5, we perform latent-space interpolation between two Gaussian noise samples. As demonstrated in Fig. 1, both SD 1.5 and NanoSD exhibit smooth and semantically consistent transitions along the interpolation path, maintaining scene layout and object semantics. This behavior suggests that NanoSD successfully retains the high-level generative structure of the original diffusion model.

1.2. Exploring Embedding-Based Similarity and Perceptual Consistency

To quantitatively assess prior preservation, we measure embedding-based similarities using CLIP encoders. If NanoSD successfully retains the generative prior of SD 1.5, its outputs should maintain proximity to the distribution generated by the original SD 1.5 model. As evidenced in Table 1, NanoSD demonstrates significantly higher similarity to SD 1.5 compared to the U-Net baseline, while approaching the self-consistency of SD 1.5 evaluated across independent sampling runs. These results indicate that NanoSD effectively preserves the generative manifold of the original SD 1.5 model.

We further assess perceptual fidelity using the Learned Perceptual Image Patch Similarity (LPIPS) [21] metric for image super-resolution. As shown in Table 1, NanoSD achieves a substantially lower perceptual distance to SD 1.5 compared to the UNet baseline. Crucially, the LPIPS difference between NanoSD and SD 1.5 is only marginally larger than the variation observed between different SD 1.5 outputs generated with distinct random seeds. This narrow margin demonstrates that NanoSD maintains strong alignment with the teacher model's output distribution despite the distillation process.

2. Real-time Image Restoration using NanoSD

2.1. Mobile ISP for Image Restoration

We address the significant challenge of developing a foundation model capable of preserving generative priors while operating efficiently on resource-constrained edge devices. Deploying such models on mobile platforms presents multiple practical obstacles, including large model sizes and high inference latency. These challenges are particularly pronounced for restoration tasks, which typically require processing high-resolution images through multiple model runs. For instance, modern smartphones often capture images at 4K×3K resolution, whereas diffusion-based models like StableSR typically generate 512×512 patches per forward pass. This discrepancy necessitates tile-based processing, where a 1000×750 low-quality input image is divided into 128×128 patches with 25% overlap, resulting in 88 patches that must be processed sequentially as shown in Fig 2. Each tile undergoes one or more diffusion steps, substantially increasing computational overhead and runtime.

2.2. Limitations of SD 1.5 for Edge Deployment

While diffusion models such as SD 1.5 offer strong generative priors, their computational and memory demands make them fundamentally unsuitable for resource-constrained devices. SD 1.5 contains approximately 829 million parameters, corresponding to about 3.3 GB of raw model weights in Full precision. Even with INT8 quantization, the footprint remains roughly 0.8 GB, exceeding the memory budgets of mobile NPUs, which must also accommodate activation buffers, intermediate tensors, runtime libraries, and system services. This alone renders SD 1.5 impractical for on-device deployment.

In practice, the situation is even more restrictive. Attempts to deploy SD 1.5 on commercial mobile chipsets failed to generate executable binaries for both Qualcomm NPUs (Android) and the Apple Neural Engine (iOS) when using FP32, INT16, or INT8 precision. Even aggressive INT4 quantization—despite its substantial accuracy degradation—yielded a model with an average latency of 116 ms per tile on the Qualcomm Snapdragon SM8750 NPU.

Recalling the tile-based pipeline described previously, a 1000×750 input image requires 88 tiles at 128×128 resolution. Thus, the end-to-end runtime of SD 1.5 becomes 10.2 seconds (88 tiles x 116 ms/tile). This latency corresponds to a *single* diffusion step, whereas restoration pipelines typically require multiple steps or sequential subnetworks (e.g., face refinement or iterative enhancement). Such runtimes are incompatible with real-time or interac-

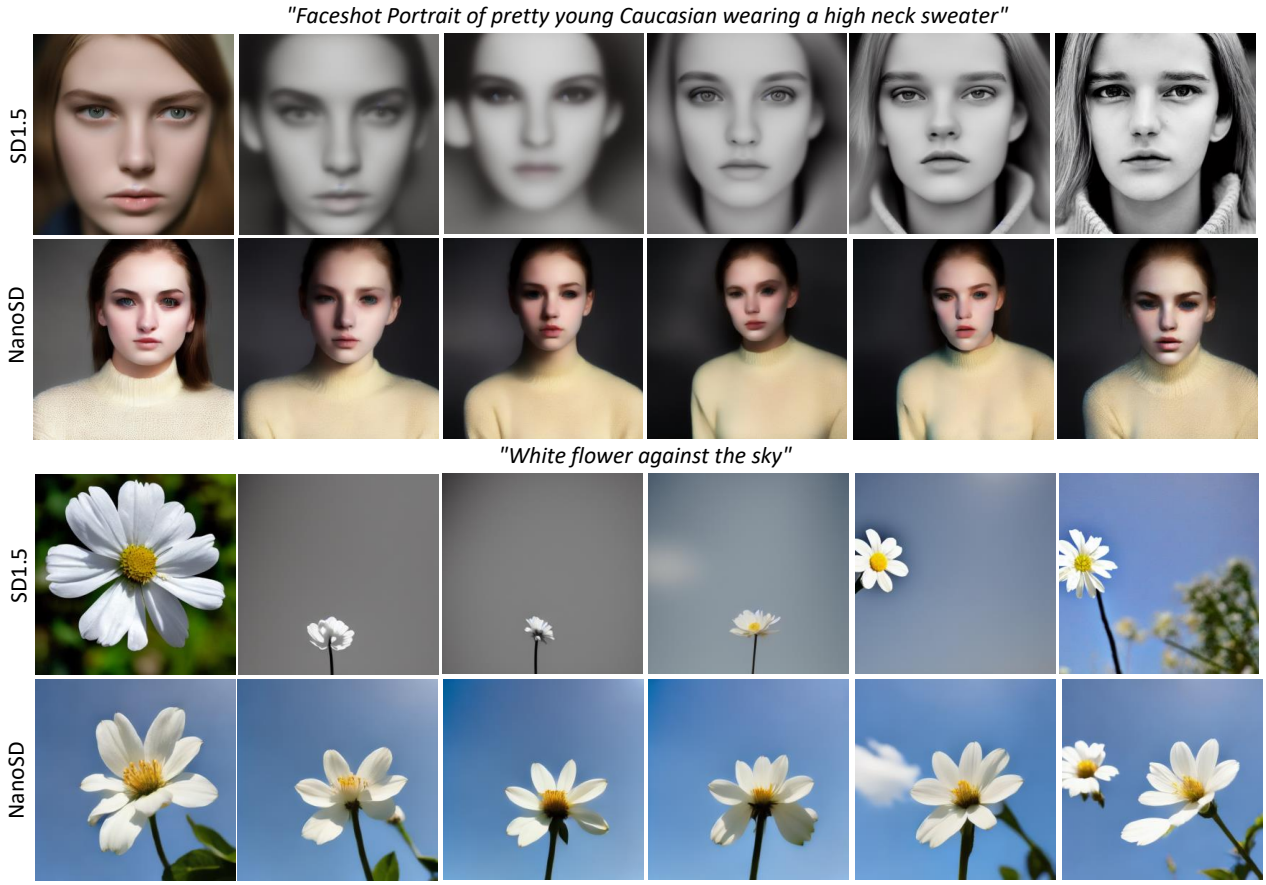

Figure 1. Latent-space interpolation between two random seeds for both SD 1.5 and NanoSD. The results demonstrate smooth transitions in the generated outputs along the interpolation trajectories for both models. This observation confirms that NanoSD preserves a well-structured latent manifold, maintaining consistency with the generative prior learned by SD 1.5.

Table 1. Perceptual distance measured using LPIPS and embedding-space similarity via CLIP embeddings between SD 1.5 outputs and other competitive models. Our results demonstrate that NanoSD maintains significantly closer proximity to SD 1.5 than the regression-based U-Net baseline. This quantitative evidence indicates robust preservation of SD 1.5's generative prior in the distilled NanoSD model.

Model Pair	LPIPS ↓	Embedding Cosine Similarity ↑
SD 1.5 vs SD 1.5 (two random runs)	0.48	0.89
NanoSD vs SD 1.5	0.57	0.84
UNet-Baseline vs SD 1.5	1.92	0.41

tive workloads. Together, these memory, compilation, and runtime constraints demonstrate why SD 1.5 is unsuitable for deployment on modern mobile devices and motivate the need for an edge-aware redesign.

2.3. Hardware-Aware Network Surgery

To bridge this gap, we perform a principled hardware-aware decomposition of the SD 1.5 U–Net. Rather than applying uniform pruning or parameter-centric compression, we restructure the network at the block level through *network surgery*. Specifically, we identify alternative block configurations—such as residual-only, reduced-attention, or hybrid arrangements—that preserve the input–output tensor shapes of the original SD 1.5 blocks. These alternatives are then individually profiled on the Qualcomm SM8750 NPU, enabling us to measure the true hardware latency of each candidate block.

This profiling reveals substantial variation in runtime among shape-compatible alternatives, with several variants offering 3–8$\times$ lower latency despite comparable parameter

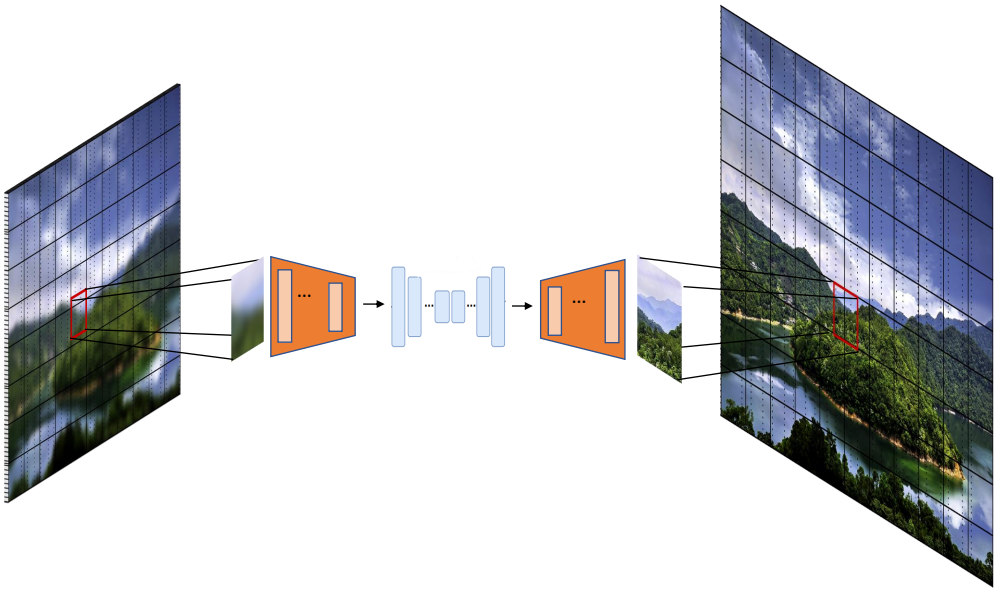

Figure 2. NanoSD employs a tiled inference strategy to enable high-resolution image restoration on edge computing platforms. The processing pipeline begins by partitioning a 1000×750 input image into 128×128 overlapping tiles (25% overlap), generating 88 total tiles. Each tile undergoes independent processing through the NanoSD model before being reassembled into a final 4K resolution output. This tile-based approach achieves two critical objectives: (1) maintaining perceptual consistency across the reconstructed image, and (2) meeting the computational constraints of mobile processors for real-time operation. The method effectively balances restoration quality with the practical requirements of edge deployment.

Table 2. Latency profiling of 30 surrogate U–Net blocks on Qualcomm SM8750 NPU (Samsung S25 Ultra). We report the measured per-block latency for all hardware-aware alternatives generated via network surgery. Although all variants preserve tensor shapes and remain accuracy-compatible after feature wise distillation, they exhibit substantial diversity in hardware cost, with several alternatives achieving 3-8× lower latency than the original SD 1.5 blocks. These measurements form the basis of the hardware-aware search space used in our Pareto optimization.

Encoder 1		**Encoder 2**		**Encoder 3**	
Alternative	**Latency (ms)**	**Alternative**	**Latency (ms)**	**Alternative**	**Latency (ms)**
R	3	R	1.79	R	1.1
RA	12.5	RA	3.7	RA	1.46
RR	3.5	RR	3	RR	1.2
Decoder 1		**Decoder 2**		**Decoder 3**	
Alternative	**Latency (ms)**	**Alternative**	**Latency (ms)**	**Alternative**	**Latency (ms)**
R	1.7	R	2.07	R	1.3
RA	16.3	RA	5.8	RA	3.4
RR	3.1	RR	3.17	RR	2.3
RRA	17.3	RRA	6.8	RRA	4.2
RAR	17.7	RAR	7.5	RAR	4.5
RRRA	19	RRRA	8.8	RRRA	5.5
RARR	19.1	RARR	7.9	RARR	5.2
RARA	39	RARA	12.8	RARA	5.5

counts. Table 2 lists all 30 surrogate blocks and their measured latencies on the SM8750 NPU in full precision. Such profiling enables us to embed hardware cost directly into the design space, avoiding reliance on FLOPs or parameter count, which correlate poorly with real-world latency due to tensor-movement and memory-access bottlenecks.

While hardware profiling identifies efficient block candidates, directly substituting them into the U–Net would disrupt SD 1.5's generative prior. To preserve fidelity, we apply *Feature-wise Generative Distillation* (FwGD) to every can-

Table 3. Cross-platform latency analysis of NanoSD family on the Apple A17 Pro Neural Engine (iOS). We evaluate all U–Net candidates from the NanoSD Pareto set—originally optimized using Qualcomm SM8750 latency data—on the Apple ANE without modification. The relative latency ordering and efficiency trends mirror those observed on SM8750, confirming that the proposed hardware-aware search produces architectures that generalize across accelerators. This consistency demonstrates the hardware-agnostic nature of our method and highlights the limitations of FLOP- or parameter-centric compression for predicting real-world edge performance.

Model	Latency (ms)	
	Qualcomm	Apple
TinySD	74	192
Hand-tuned	53	133
NanoSD 1	41	82
NanoSD 2	27	38
NanoSD 3	24	34
NanoSD 4	20	31
NanoSD 5	12	20
NanoSD 6	28	68
NanoSD 7	27	66

didate block: each surrogate is trained to match the output distribution of its corresponding SD 1.5 teacher block under identical inputs. The structural compatibility between the base block and each hardware-aware alternative ensures the path for knowledge transfer to compact, local modules, enabling massively parallel and computationally lightweight training.

FwGD provides two essential benefits:

- **Accuracy preservation.** Each block independently inherits SD 1.5's generative behavior, allowing arbitrary combinations of surrogates to form valid U–Nets without catastrophic feature drift.
- **Search-space validity.** By distilling all block variants, the search space contains only architectures that are simultaneously hardware-efficient and accuracy-sufficient.

Together, hardware-aware network surgery and FwGD yield a search space that is structurally valid, hardware-aligned, and generative-prior-preserving—properties crucial for downstream optimization.

2.4. Why Hardware Awareness Is Essential?

If the search space were constructed solely using parameter reduction, FLOPs minimization, or uniform pruning, the resulting architectures would not necessarily yield improvements in *true* edge latency. In hierarchical U–Nets, inner blocks operate at low spatial resolutions, so reducing their parameters has minimal effect on runtime. Conversely, even small modifications to early, high-resolution stages significantly reduce latency due to the cost of propagating large activation maps. This mismatch between compute-centric metrics and latency-critical behavior explains why SD 1.5 cannot simply be "compressed" into an edge model.

By profiling block alternatives directly on NPU hardware, our method ensures that latency-critical regions receive appropriate structural simplifications. Combined with FwGD, this yields a hardware-aware and accuracy-aligned search space from which Bayesian optimization extracts Pareto-optimal U–Nets, ultimately producing the NanoSD architecture. This synergy between hardware profiling, network surgery, and block-level distillation fundamentally differentiates our method from FLOP-centric or parameter-centric reduction techniques.

2.5. Generic Nature of the Proposal

To assess the generality of our approach, we replicated the hardware profiling procedure on a second, architecturally distinct platform: the Apple A17 Pro Neural Engine (ANE) on iOS. Although the network surgery and Pareto optimization were performed using latency measurements from the Qualcomm SM8750 NPU on the Samsung S25 Ultra, we evaluated the resulting NanoSD family on the ANE without modification. As shown in Table 3, the latency trends on the ANE closely mirror those observed on the SM8750: NanoSD variants consistently outperform the SD 1.5 baseline and manually designed alternatives despite differences in compiler stacks, operator scheduling, and memory hierarchies across the two platforms. This consistency demonstrates that the proposed method is inherently hardware-agnostic: rather than tailoring a model to a single device, it constructs a search space aware of device characteristics while remaining compatible with any backend NPU. These results reinforce that FLOPs and parameter count alone are insufficient predictors of on-device performance, and that explicit hardware-aware decomposition is essential for designing diffusion models suitable for diverse edge environments.

2.6. Generality to Transformer-based Diffusion Models and Quantization.

While the main paper focuses on U-Net backbones due to their practicality for real-time edge deployment, the proposed framework is architecture-agnostic. Our method operates at the level of shape-preserving block variants and feature-wise generative distillation, which applies equally to convolutional, attention-based, and MLP-based blocks. As a proof-of-concept, we conducted an ablation on a vanilla DiT model by replacing the final decoder MLP block ($4\times$ expansion) with a lightweight surrogate ($2\times$ expansion) aligned via our block-wise distillation. Evaluated using Kernel Inception Distance (KID), we obtained 0.06 ± 0.002 with minimal visual degradation, demonstrating direct applicability to transformer-based diffusion mod-

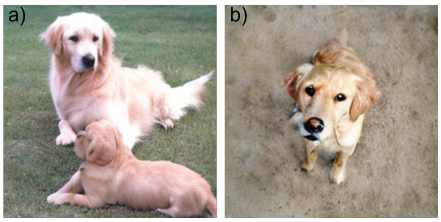

Figure 3. (a) Baseline DiT. (b) Distilled DiT (INT8 quantization).

els. Regarding quantization, we primarily report w8a16 since it is the most stable and widely supported configuration on mobile NPUs. In the DiT ablation, we additionally quantized the distilled surrogate block to INT8 while keeping the rest of the network in FP32, illustrating mixed-precision deployment within diffusion pipelines (see Fig. 3 for image quality comparison). Full INT4/FP4 end-to-end quantization constitutes a promising future direction.

2.7. Edge efficiency of NanoSD

We conduct a systematic evaluation of NanoSD's computational efficiency to assess its viability for edge deployment in high-resolution image restoration applications. Our quantitative analysis reveals that NanoSD achieves inference latencies compatible with real-time operational requirements on mobile platforms. Specifically, the model processes a 128×128 input tile to produce a 512×512 output in approximately 20 ms when running on mobile neural processing units (NPUs). For processing 4000×3000 image, the total end-to-end latency measures of about 1.8 seconds on equivalent hardware, accounting for all 88 required tile evaluations. These empirical results provide compelling evidence that diffusion-based restoration can be effectively implemented on-device, satisfying the stringent computational constraints of mobile platforms while eliminating the need for cloud-based processing.

3. Bayesian Optimization with EHVI

Unlike prior compression approaches that operate through progressive layer removal or step distillation on a fixed backbone, our method reformulates Stable Diffusion as a hardware-aware space of shape-preserving block variants, each profiled on target NPUs and aligned via modular generative distillation.

We cast the selection of an efficient U–Net backbone from the discrete hardware-aware search space as a multi-objective black-box optimization problem. Let $\mathcal{Z}$ denote the discrete set of feasible architectures induced by stage-wise block choices. For any architecture $\mathbf{z} \in \mathcal{Z}$ we evaluate two primary objectives:

$$f_{\mathrm{FID}}(\mathbf{z}) = \mathrm{FID}\Big(\hat{X}(\mathbf{z}), X_{\mathrm{SD1.5}}\Big), \tag{1}$$

$$f_{\mathrm{param}}(\mathbf{z}), \qquad f_{\mathrm{latency}}(\mathbf{z}). \tag{2}$$

where f_{FID}, the first objective (f_1), is the teacher-aligned FID measured against SD 1.5 outputs, the second objective (f_2) is either $f_{latency}$, the on-device latency or f_{param}, the parameter count, depending on the search iteration.

Direct optimization over $\mathcal{Z}$ is expensive because each evaluation of f_1 requires many diffusion samples and FID computation. We therefore employ Bayesian Optimization (BO) with Expected Hypervolume Improvement (EHVI) [8] as the acquisition function to efficiently explore $\mathcal{Z}$. To handle the discrete nature of $\mathcal{Z}$ we adopt a two-step relaxed-and-project approach: (1) represent architecture choices by a continuous proxy $\mathbf{x} \in [0,1]^d$ (here d is the number of design dimensions), (2) map $\mathbf{x}$ to the nearest feasible discrete architecture via a deterministic projection $\phi : [0,1]^d \to \mathcal{Z}$.

The BO loop proceeds as follows. We fit independent Gaussian Process (GP) surrogates to the observed values of each objective over previously evaluated architectures. EHVI is computed with respect to the current Pareto set and used to propose the next continuous candidate $\mathbf{x}^*$. The projection $\mathbf{z}^* = \phi(\mathbf{x}^*)$ yields a discrete architecture which is assembled from the distilled surrogate blocks and then evaluated to obtain $(f_1(\mathbf{z}^*), f_2(\mathbf{z}^*))$. The GP models are updated and the Pareto set is expanded accordingly. Iteration continues until a predefined budget of evaluations is exhausted, resulting the final Pareto set as shown in Figure 2 in the main paper.

4. VAE Distillation

Our objective is to obtain a fully edge-optimized latent diffusion model by compressing all major components of Stable Diffusion 1.5-the U-Net, VAE encoder, and VAE decoder-while preserving compatibility with the teacher model's generative behavior. Starting from a pretrained SD 1.5 pipeline, we first distill the U-Net, then distill the VAE encoder and decoder with the U-Net frozen, and finally refine the assembled pipeline for end-to-end consistency.

Table 4 provides an overview and includes the distilled architectures of both the encoder and decoder. The proposed student VAE is highly parameter- and latency-efficient: its encoder and decoder contain only ∼2 M and ∼1.3 M parameters (10 ms and 8 ms FP16 latency, respectively), compared to the much larger SD,1.5 VAE. This reduction is achieved by using fixed-width (64-channel) Tiny ResNet blocks and shallow lightweight up/downsampling stages instead of the widening 64→128→256→512 ResNet blocks used in the teacher.

Table 4. Architectural comparison of the Teacher VAE (SD 1.5) and the proposed Student VAE. Both models are shown using a unified notation with identical spatial dimensions. Only architectural differences are listed. ResNetBlockTiny refers to a lightweight residual block that preserves the Conv–Norm–Activation–Conv pattern of a standard ResNet block but replaces the full 3×3 convolutions with parameter–efficient variants (e.g. depthwise–separable convs) operating at a fixed 64-channel width. This makes each block an order of magnitude smaller than the full ResNet blocks in the SD 1.5 encoder.

Teacher VAE (SD 1.5)	**Student VAE (Proposed)**
Encoder Architecture (Input: $3 \times H \times W$)	
Stem: Conv(3→64), ResNetBlock ×2. Resolution preserved.	**Stem:** Conv(3→64), **ResNetBlockTiny ×1**. Resolution preserved.
Down Block 1: ResNetBlock ×2 @ 64 ch, stride-2 Conv $\rightarrow H/2 \times W/2$.	**Down Block 1: ResNetBlockTiny ×3** @ 64 ch, stride-2 Conv $\rightarrow H/2 \times W/2$.
Down Block 2: ResNetBlock ×2 @ 128 ch, stride-2 Conv $\rightarrow H/4 \times W/4$.	**Down Block 2: ResNetBlockTiny ×3** @ 64 ch, stride-2 Conv $\rightarrow H/4 \times W/4$.
Down Block 3: ResNetBlock ×2 @ 256 ch, stride-2 Conv $\rightarrow H/8 \times W/8$.	**Down Block 3: ResNetBlockTiny ×3** @ 64 ch, stride-2 Conv $\rightarrow H/8 \times W/8$.
Middle/Bottleneck: ResNetBlock ×2 @ 512 ch.	**Middle/Bottleneck:** (no dedicated bottleneck; final stage already at 64 ch).
Output projection: Conv→8 ch.	**Output projection:** Conv→4 ch.
Decoder Architecture (Input: $z \in \mathbb{R}^{4\times(H/8)\times(W/8)}$)	
Stem: Conv(4→512).	**Stem:** Conv(4→64).
Middle/Bottleneck: ResNetBlock ×2 @ 512 ch.	**Middle/Bottleneck:** ResNetBlock ×1 @ 64 ch.
Up Block 1: Upsample×2, ResNetBlock ×2 (512→256) $\rightarrow H/4 \times W/4$.	**Up Block 1:** ResNetBlock ×2 @ 64 ch, Upsample×2 $\rightarrow H/4 \times W/4$.
Up Block 2: Upsample×2, ResNetBlock ×2 (256→128) $\rightarrow H/2 \times W/2$.	**Up Block 2:** ResNetBlock (→128 ch), Upsample×2 $\rightarrow H/2 \times W/2$.
Up Block 3: Upsample×2, ResNetBlock ×2 (128→64) $\rightarrow H \times W$.	**Up Block 3:** ResNetBlock @ 128 ch, Upsample×2 $\rightarrow H \times W$.
Final projection: Conv(64→3).	**Final projection:** Conv(128→3).

4.1. Preliminaries

Let E_t, D_t, and U_t denote the teacher VAE encoder, VAE decoder, and diffusion U-Net. The encoder outputs posterior parameters

$$(\mu_t(x), \log \sigma_t^2(x)) = E_t(x). \tag{3}$$

Latent sampling follows

$$z_t = \mu_t(x) + \sigma_t(x) \odot \epsilon, \qquad \epsilon \sim \mathcal{N}(0, I). \tag{4}$$

Scaled latents are fed into the U-Net:

$$\tilde{z}_t = \alpha z_t, \qquad \alpha = 0.18215. \tag{5}$$

The U-Net predicts noise conditioned on time step t and text embedding c:

$$\epsilon_t = U_t(\tilde{z}_t, t, c). \tag{6}$$

Our goal is to obtain student modules E_s, D_s, and U_s such that the resulting pipeline remains distributionally aligned with the teacher.

4.2. Stage I: U-Net Distillation

The student U-Net U_s is trained via the proposed formulation (see Figure 2 in main paper) for edge efficiency, resulting in the NanoSD U-Net. After this stage, U_s is frozen for the remainder of training.

4.3. Stage II: Encoder Distillation

The student encoder outputs

$$(\mu_s(x), \log \sigma_s^2(x)) = E_s(x), \tag{7}$$

and is trained to match the teacher encoder via

$$\mathcal{L}_{\text{latent}} = \|\mu_t(x) - \mu_s(x)\|_2^2 + \|\sigma_t(x) - \sigma_s(x)\|_2^2. \tag{8}$$

A weak KL regularization term is included:

$$\mathcal{L}_{\text{KL}} = D_{\text{KL}}(q_s(z \mid x) \| \mathcal{N}(0, I)). \tag{9}$$

The encoder objective is:

$$\mathcal{L}_E = \mathcal{L}_{\text{latent}} + \beta\, \mathcal{L}_{\text{KL}}, \qquad \beta \in [10^{-6}, 10^{-4}]. \tag{10}$$

4.4. Stage III: Decoder Distillation

Given teacher latents z_t, the teacher and student reconstructions are:

$$\hat{x}_t = D_t(z_t), \qquad \hat{x}_s = D_s(z_t). \tag{11}$$

Teacher-guided reconstruction is enforced via

$$\mathcal{L}_{\text{recon}} = \|\hat{x}_t - \hat{x}_s\|_1. \tag{12}$$

A perceptual loss aligns higher-level features:

$$\mathcal{L}_{\text{perc}} = \|\phi(\hat{x}_t) - \phi(\hat{x}_s)\|_2^2. \tag{13}$$

To ensure decoder robustness to student latents z_s, we include

$$\mathcal{L}_{\text{AE}} = \|x - D_s(z_s)\|_1. \tag{14}$$

The decoder objective is thus

$$\mathcal{L}_D = \mathcal{L}_{\text{recon}} + \lambda_{\text{perc}}\mathcal{L}_{\text{perc}} + \lambda_{\text{AE}}\mathcal{L}_{\text{AE}}. \tag{15}$$

4.5. Stage IV: Joint VAE Refinement

Once E_s and D_s are individually trained, we jointly refine them using

$$\mathcal{L}_{\text{VAE}} = \lambda_1\|x - D_s(E_s(x))\|_1 + \lambda_2\mathcal{L}_{\text{perc}} + \lambda_3 D_{\text{KL}}(q_s\|\mathcal{N}(0, I)). \tag{16}$$

4.6. Stage V: End-to-End Diffusion Alignment

To compensate for latent shift introduced by VAE compression, we lightly fine-tune U_s while keeping E_s and D_s frozen:

$$\mathcal{L}_{\text{align}} = \|U_s(\alpha z_s + \sigma_t\epsilon, t, c) - U_t(\alpha z_t + \sigma_t\epsilon, t, c)\|_2^2. \tag{17}$$

This multi-stage procedure yields a compact, edge-efficient latent diffusion model that preserves the generative behavior of SD 1.5 while enabling practical on-device deployment.

5. Implementation Details

Hardware for Latency Profiling. All hardware-aware latency measurements were performed on a Qualcomm Snapdragon SM8750 NPU deployed on a Samsung Galaxy S25 Ultra. Each of the 30 block alternatives was compiled using Qualcomm SNPE with INT8 weights and FP16 activations, and latency was averaged over 256 runs after discarding the first 20 warm-up iterations. To assess generality across heterogeneous hardware, the final NanoSD U-Net family was also profiled on the Apple A17 Pro Neural Engine using CoreML Tools on iOS 17, where a consistent latency ranking was observed (Table 3).

Dataset for Distillation. Feature-wise Generative Distillation, VAE distillation, and end-to-end alignment were all conducted using a representative subset of approximately 1 million image-text pairs sampled from a LAION-style corpus. This subset was stratified across prompt categories and image types to ensure coverage equivalent to the teacher model's operating distribution. A held-out validation subset of 50,000 samples was used for early stopping and monitoring drift.

Feature-wise Generative Distillation of 30 Surrogates. Each of the 30 U-Net surrogate blocks was distilled independently against the corresponding SD 1.5 teacher block. We used AdamW (learning rate 1×10^{-4}, weight decay 0.01), batch size 128, cosine decay schedule with 2k warmup steps, and 15k–40k training iterations depending on block complexity. The total compute budget for all 30 surrogates was approximately 360 A100 GPU-hours. Parallel training across multiple GPUs allowed all surrogates to be distilled simultaneously.

Bayesian Optimization Configuration. Bayesian optimization was applied separately for the two bi-objective settings: (i) taFID vs. edge latency, and (ii) taFID vs. parameter count. We used a 6-dimensional continuous relaxation of the discrete search space, projected to feasible architectures via a deterministic thresholding map. Independent Gaussian Processes with Matérn-5/2 kernels were fitted to both objectives. The acquisition function was Expected Hypervolume Improvement (EHVI). We used 15 random initial samples and 120 BO iterations per run, and performed 5 independent runs with different seeds. The total compute cost for BO evaluation-dominated by taFID computation-was approximately 100 A100 GPU-hours.

VAE Encoder and Decoder Distillation. After selecting the NanoSD U-Net, we distilled the VAE encoder and decoder using the same 1M-sample dataset. The encoder was trained with AdamW (learning rate 5×10^{-5}, batch size 64) for 40k iterations, minimizing latent-matching and KL regularization losses with $\beta \in [10^{-6}, 10^{-4}]$. The decoder was trained for 80k–120k iterations with AdamW (learning rate 1×10^{-4}), using reconstruction, perceptual, and autoencoding consistency losses. Combined encoder+decoder distillation required approximately 220 A100 GPU-hours.

End-to-End Diffusion Alignment. To compensate for the mild latent shift introduced by VAE compression, we performed a lightweight end-to-end noise-prediction alignment stage. The NanoSD U-Net was fine-tuned with a small learning rate (1×10^{-6}) for 20k–40k steps, with the VAE frozen. We used batch size 32, random timesteps, and kept

the text encoder frozen. This stage required an additional 70 A100 GPU-hours.

Total Compute. Across all stages (FwGD, BO evaluations, VAE distillation, and T2I end-to-end alignment), the complete NanoSD pipeline required approximately 750 A100 GPU-hours-substantially lower than training or distilling a full SD 1.5 model, while enabling hardware-aware optimization tailored to edge devices.

6. Experimental Results

We present a detailed experimental and ablation analysis of various low-level vision tasks using the proposed NanoSD.

6.1. Ablation on Removing `E4--Mid--D4` Blocks.

Prior work has shown that the highest-resolution stages of the U-Net dominate per-layer computational cost, whereas the lowest-resolution stages contribute disproportionately to the memory footprint with a limited effect on perceptual quality. Motivated by this observation, our default search space removes the encoder-4, mid, and decoder-4 blocks (`E4--Mid--D4`). To validate this choice, we reintroduced these three blocks (configured as `E4: R` $\rightarrow$ `Mid: RAR` $\rightarrow$ `D4: RR`) into one of our baseline models, `NanoSD 1` (see Table 1 of the main paper). The original model has a latency of 41 ms and 309 M parameters. After adding `E4--Mid--D4`, the parameter count increases substantially to 565 M, while latency increases only marginally to 46 ms due to the small spatial resolutions processed by these blocks.

We compare the pre-finetuning image quality of the two variants in Figure 4. Despite the large increase in parameters and memory footprint, the improvement in visual quality is negligible. This confirms that the cost of including `E4--Mid--D4` is not justified for edge-efficient models, and supports our decision to exclude these blocks from the search space.

6.2. Scratch Training vs. Distillation.

To isolate the value of diffusion-based distillation, we trained the identical NanoSD architecture for $4\times$ super-resolution under two settings: (i) training from scratch and (ii) SD 1.5 block-wise distilled initialization followed by identical fine-tuning. As shown in Table 7 and Fig. 5, the distilled NanoSD consistently outperforms the scratch-trained model. This confirms that architectural efficiency alone cannot recover the generative prior inherited from SD 1.5; distillation provides a substantially better preservation of generative behavior.

Figure 4. Ablation demonstrating the effect of reintroducing the `E4--Mid--D4` blocks into `NanoSD 1`. Column 1: NanoSD 1 and Column 2: NanoSD 1 + `E4--Mid--D4`. Although these additional blocks significantly increase the parameter count (309 M $\rightarrow$ 565 M), the latency impact is minimal (41 ms $\rightarrow$ 46 ms), since they operate at low spatial resolution. However, as illustrated, the pre-finetuning image quality is nearly unchanged, showing that the large memory overhead provides no practical benefit. This justifies eliminating `E4--Mid--D4` from our search space for improved edge efficiency.

Figure 5. (a) Input, (b) NanoSD (Scratch), (c) NanoSD (Distilled).

Table 5. Quantitative comparison of different methods on DIV-2K Val [1]to dataset. The best, second-best and third-best results are highlighted in red, blue, and green colors, respectively.

Method	PSNR↑	SSIM↑	LPIPS↓	FID↓	NIQE↓	MUSIQ↑	Steps	MACs (G)	Para. (M)
StableSR [10]	23.26	0.572	0.311	24.44	4.75	65.92	200	79940	1410
DiffBIR [7]	23.64	0.564	0.352	30.72	4.70	65.81	50	24234	1717
SeeSR [16]	23.68	0.604	0.319	25.90	4.81	68.67	50	65857	2524
ResShift [19]	24.65	0.618	0.334	36.11	6.82	61.09	15	5491	119
SinSR [13]	24.41	0.601	0.324	35.57	6.02	62.82	1	2649	119
OSEDiff [15]	23.72	0.610	0.294	26.32	4.71	67.97	1	2265	1775
S3Diff [20]	23.52	0.594	0.258	19.66	4.74	68.01	1	2621	1327
TinySR [4]	-	0.572	0.279	22.94	4.15	69.90	1	427	341
Edge-SD-SR [5]	24.10	0.617	0.249	25.37	-	69.58	1	-	169
PocketSR [9]	23.85	0.601	0.280	25.25	4.415	66.38	1	225	146
AdcSR [3]	23.74	0.601	0.285	25.52	4.36	68.00	1	496	456
Nano-OSEDiff (Ours)	24.29	0.628	0.296	27.46	4.92	66.41	1	340	448
Nano-S3Diff (Ours)	23.13	0.573	0.278	22.34	4.09	70.44	1	285	318

Table 6. Quantitative comparison of different methods on real-world datasets. The best, second-best and third-best results are highlighted in red, blue, and green colors, respectively.

Datasets	DRealSR [14]					RealSR [2]				
Method	PSNR↑	SSIM↑	LPIPS↓	FID↓	MUSIQ↑	PSNR↑	SSIM↑	LPIPS↓	FID↓	MUSIQ↑
SinSR [13]	28.36	0.751	0.366	170.5	55.33	26.28	0.734	0.318	135.9	60.80
OSEDiff [15]	27.92	0.783	0.296	135.3	64.65	25.15	0.734	0.292	123.4	69.09
S3Diff [20]	27.39	0.746	0.312	119.2	64.16	25.19	0.731	0.270	110.3	67.92
Edge-SD-SR [5]	-	-	0.292	-	55.66	-	-	0.278	-	65.20
AdcSR [3]	28.10	0.772	0.304	134.0	66.26	25.47	0.730	0.288	118.4	69.90
TinySR [4]	27.48	0.745	0.311	146.7	65.36	24.79	0.717	0.280	118	69.78
PocketSR [9]	28.05	0.767	0.296	-	63.85	25.47	0.733	0.271	-	67.07
Nano-OSEDiff	29.01	0.808	0.276	134.6	58.84	26.19	0.746	0.272	122.7	65.29
Nano-S3Diff	26.96	0.735	0.321	127.1	67.83	24.47	0.701	0.281	116.8	70.27

Table 7. Comparison of scratch training and distillation training.

Dataset	Model	PSNR↑	LPIPS↓	FID↓	MUSIQ↑
Div2K	NanoSD	25.06	0.381	23.34	67.03
	NanoSD (Scratch)	23.94	0.438	27.52	61.81
RealSR	NanoSD	26.05	0.331	127.76	69.90
	NanoSD (Scratch)	24.51	0.373	143.17	60.19

Table 8. Quantitative comparison on CelebA-Test dataset.

Method	IDS ↑	AFICS ↑	LMD ↓
DiffBIR	0.657	0.695	5.104
OSDFace	0.648	0.712	5.286
Nano-DiffBIR (130M)	0.564	0.585	7.342
Nano-OSDFace (130M)	0.559	0.577	7.861
Nano-DiffBIR (Ours)	0.644	0.689	5.231
Nano-OSDFace (Ours)	0.651	0.708	5.172

6.3. Edge Efficient OSEDiff and S3Diff for Single Image Super-resolution

This paper aims to enable real-time super-resolution by effectively harnessing the rich generative priors inherent in pre-trained diffusion models. We validate the efficacy of the proposed foundation model for edge-efficient SR tasks by integrating NanoSD with prominent one-step latent diffusion frameworks, namely OSEDiff and S3Diff.

OSEDiff employs LoRA layers integrated into the VAE encoder and diffusion UNet to perform super-resolution, thereby preserving the prior knowledge of the T2I model. We substitute the latent diffusion model in OSEDiff with NanoSD and fine-tune the asymmetric VAE encoder and UNet using LoRA, setting the rank to 4. Training follows the VSD framework consistent with OSEDiff. In contrast, S3Diff incorporates a pre-trained degradation estimation network to leverage rich content information from LR images. Its diffusion UNet is trained with a degradation-guided LoRA module and adversarial distillation for one-step SR. We integrate NanoSD into S3Diff, configuring rank parameters for the VAE encoder and diffusion UNet as 16 and 32, respectively, and adhere to S3Diff's training protocols.

Table 5 and Table 6 present quantitative comparisons of super-resolution methods across three datasets. Both pro-

Table 9. Quantitative comparison of different methods on various real-world datasets. The best, second-best and third-best results are highlighted in red, blue, and green colors, respectively.

Method	Wider-Test [22]			LFW-Test [6]			WebPhoto-Test [12]		
	MUSIQ↑	NIQE↓	FID↓	MUSIQ↑	NIQE↓	FID↓	MUSIQ↑	NIQE↓	FID↓
PGDiff [17]	68.13	3.93	35.86	71.24	4.01	41.20	68.59	3.99	86.95
DifFace [18]	64.90	4.23	37.09	69.61	3.90	46.12	65.11	4.24	79.55
DiffBIR [7]	75.32	5.59	35.34	76.42	5.67	40.32	72.27	6.00	91.83
OSEDiff [15]	70.55	4.93	50.27	73.40	4.71	57.80	72.59	5.26	117.5
OSDFace [11]	74.60	3.77	34.64	75.35	3.87	51.04	73.93	3.98	84.59
Nano-DiffBIR (Ours)	74.88	5.73	35.13	76.02	5.81	40.45	72.99	5.89	90.48
Nano-OSDFace (Ours)	74.32	3.96	35.19	75.28	3.81	53.21	74.96	4.01	85.03

posed models demonstrate robust performance and computational efficiency. Specifically, they exhibit the second-lowest MACs among evaluated approaches, enabling real-time processing and edge deployment. Nano-S3Diff achieves top NIQE and MUSIQ scores, along with second-best FID results, reflecting superior perceptual quality. Meanwhile, Nano-OSEDiff maintains competitive fidelity, yielding the superior PSNR and SSIM performance over Edge-SD-SR, PocketSR, and AdcSR while surpassing more computationally intensive methods.

Fig. 6 and Fig. 7 present qualitative comparisons between different super-resolution methods. Our proposed approaches demonstrate consistent performance in preserving image fidelity and reconstructing fine details. In the first example, both Nano-OSEDiff and Nano-S3Diff accurately reconstruct the striped pattern, while AdCSR generates artifacts and unrealistic textures. The second example shows that our methods effectively restore person details, outperforming AdcSR in terms of both face restoration, structure recovery and detail preservation. These visual results confirm that our edge-efficient models achieve high-quality reconstructions while maintaining rich textural details across various image contexts.

6.4. Edge Efficient OSDFace for Face Restoration

This section evaluates the applicability of our proposed model for face restoration tasks. To demonstrate its effectiveness, we integrate NanoSD into the OSDFace framework, which represents the current state-of-the-art one-step diffusion approach for face restoration.

OSDFace acquires comprehensive facial prior features to guide the diffusion process. These priors are generated by training a Visual Representation Embedder (VRE) on low-quality (LQ) facial images. This prior knowledge is integrated into the diffusion UNet using cross-attention layers, with model fine-tuning accomplished via LoRA. To optimize inference speed, NanoSD is incorporated into the OSDFace framework. This implementation retains VRE for prompt embedding construction and employs LoRA-based UNet fine-tuning at a rank of 32. Table 9 and Fig. 8 present comparisons of various methods on three real-world datasets, where our method consistently delivers competitive results. These findings demonstrate that Nano-OSDFace achieves substantial improvements in efficiency while maintaining performance parity with leading face restoration approaches.

6.5. Identity Preservation Analysis

For face restoration, we additionally report identity-specific metrics. We provide Identity Similarity (IDS) and Average Face ID Cosine Similarity (AFICS). Table 8 demonstrates that the proposed NanoSD achieves identity preservation comparable to original SD 1.5-based restoration method. Notably, the significant improvement over the extremely lightweight variants demonstrates that NanoSD achieves an effective accuracy–latency–model size trade-off without inducing hallucinated identity shifts.

6.6. Accelerating Diff-Plugin for Image Restoration with Generative Diffusion Prior

The Diff-Plugin framework employs task-specific priors that combine task guidance and input image spatial information. These priors enable pre-trained diffusion models to address low-level vision tasks while preserving content fidelity. Central to this approach is a lightweight Task-Plugin module, which comprises two components: the Task-Prompt Branch (TPB) for task-specific direction and the Spatial Complement Branch (SCB) for enhancing output fidelity through visual guidance. In our implementation, we substitute the original latent diffusion backbone with NanoSD. Task-specific priors extracted by the Task-Plugin are injected into NanoSD's ResNet and Cross-Attention blocks. Similar to the standard Diff-Plugin framework, the module is optimized using a denoising loss to effectively integrate task-specific guidance into the diffusion process. Fig. 9 and Fig. 10 illustrate the enhanced performance of Nano-Diff-Plugin across four challenging low-level vision tasks. Visual comparisons reveal that our lightweight approach delivers consistent results across all evaluated tasks.

Figure 6. Qualitative comparisons of different SR methods. Please zoom in for a better view

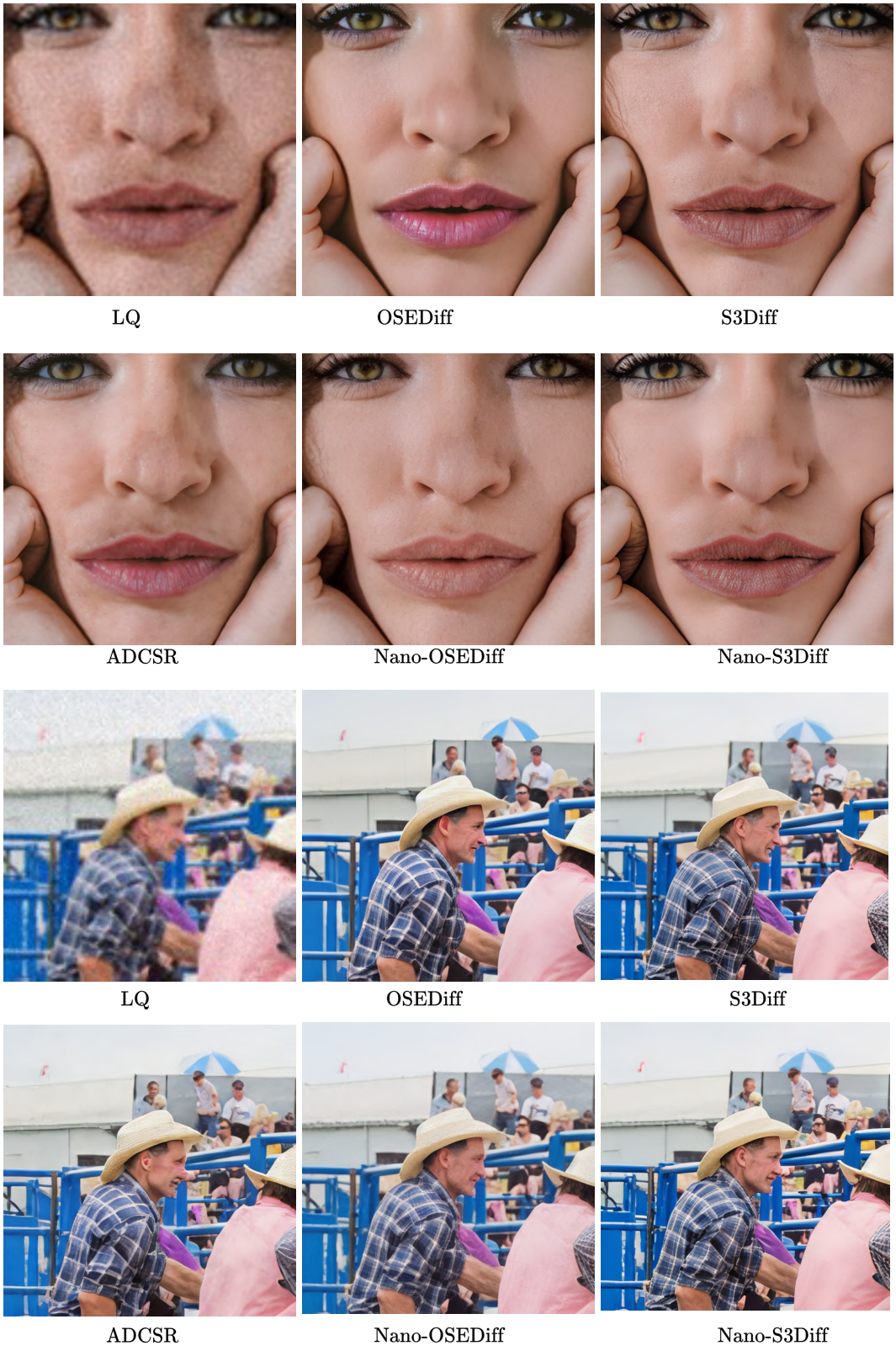

Figure 7. Qualitative comparisons of different SR methods. Please zoom in for a better view

Figure 8. Qualitative comparisons of different FR methods.

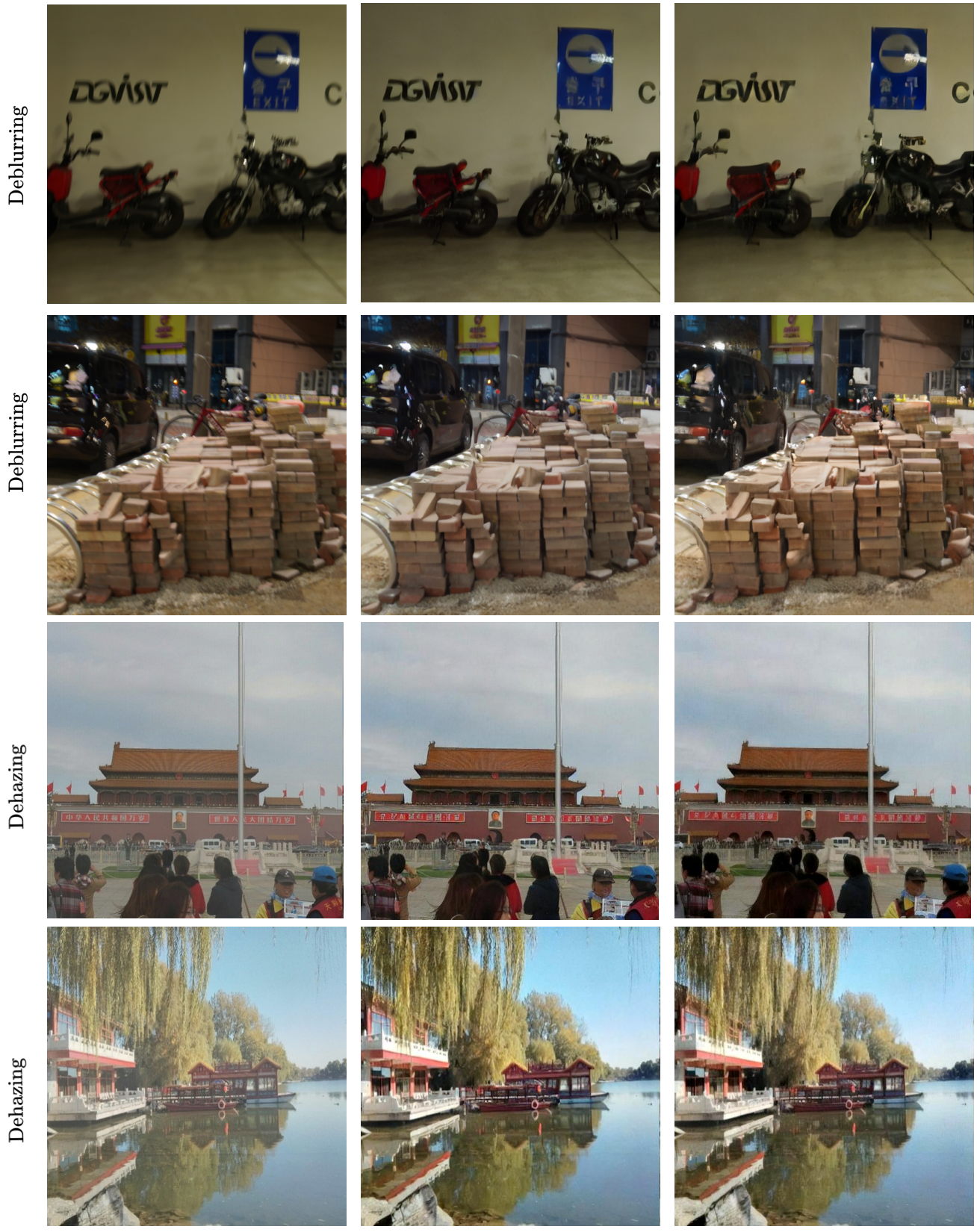

Figure 9. Qualitative comparisons of different methods for various restoration tasks

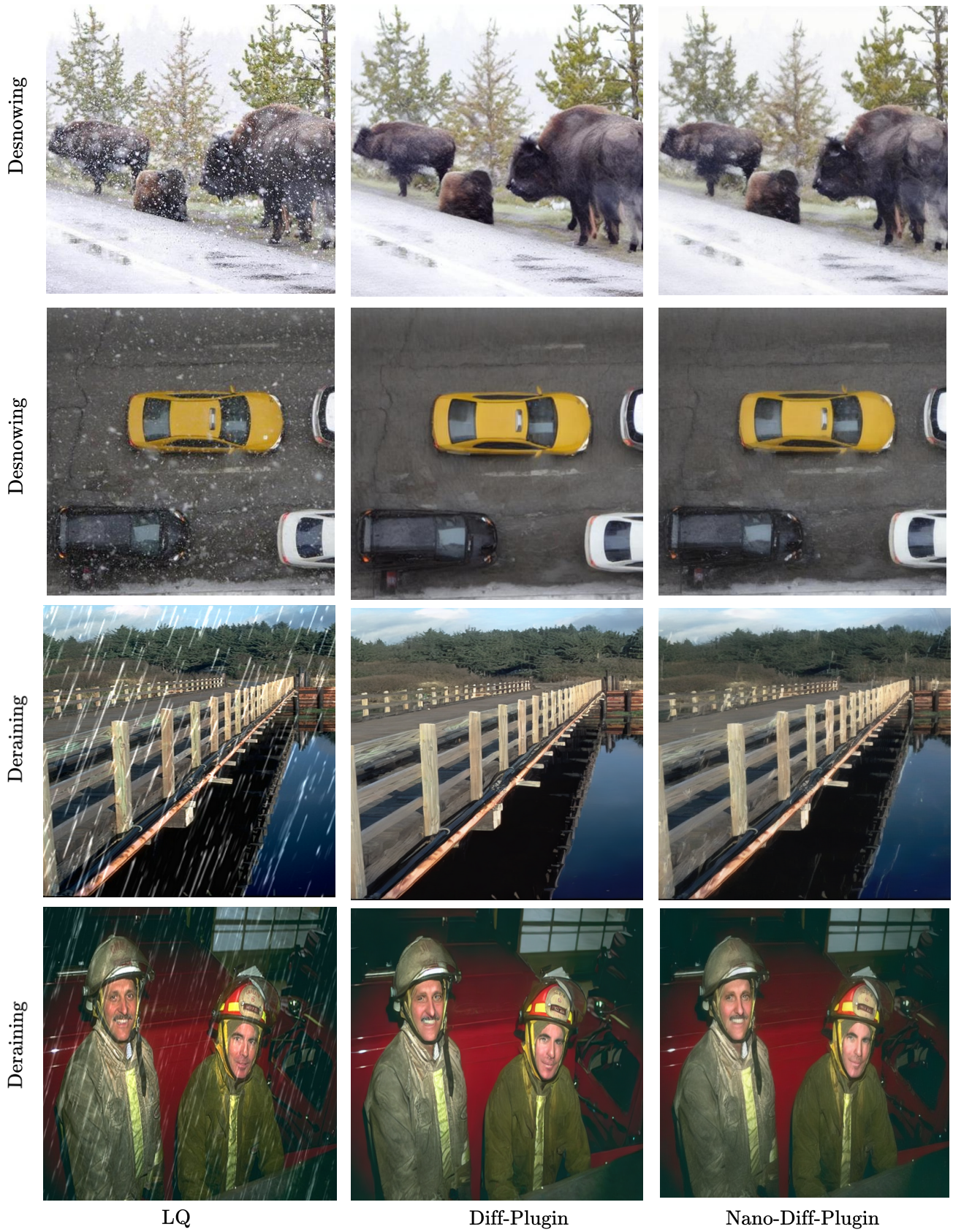

Figure 10. Qualitative comparisons of different methods for various restoration tasks

References

[1] Eirikur Agustsson and Radu Timofte. Ntire 2017 challenge on single image super-resolution: Dataset and study. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pages 126–135, 2017. 9

[2] Jianrui Cai, Hui Zeng, Hongwei Yong, Zisheng Cao, and Lei Zhang. Toward real-world single image super-resolution: A new benchmark and a new model. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 3086–3095, 2019. 9

[3] Bin Chen, Gehui Li, Rongyuan Wu, Xindong Zhang, Jie Chen, Jian Zhang, and Lei Zhang. Adversarial diffusion compression for real-world image super-resolution. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 28208–28220, 2025. 9

[4] Linwei Dong, Qingnan Fan, Yuhang Yu, Qi Zhang, Jinwei Chen, Yawei Luo, and Changqing Zou. Tinysr: Pruning diffusion for real-world image super-resolution. *arXiv preprint arXiv:2508.17434*, 2025. 9

[5] Isma Hadji, Mehdi Noroozi, Victor Escorcia, Anestis Zaganidis, Brais Martinez, and Georgios Tzimiropoulos. Edge-sd-sr: Low latency and parameter efficient on-device super-resolution with stable diffusion via bidirectional conditioning. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 12789–12798, 2025. 9

[6] Gary B Huang, Marwan Mattar, Tamara Berg, and Eric Learned-Miller. Labeled faces in the wild: A database forstudying face recognition in unconstrained environments. In *Workshop on faces in'Real-Life'Images: detection, alignment, and recognition*, 2008. 10

[7] Xinqi Lin, Jingwen He, Ziyan Chen, Zhaoyang Lyu, Bo Dai, Fanghua Yu, Yu Qiao, Wanli Ouyang, and Chao Dong. Diffbir: Toward blind image restoration with generative diffusion prior. In *European conference on computer vision*, pages 430–448. Springer, 2024. 9, 10

[8] Jasper Snoek, Hugo Larochelle, and Ryan P. Adams. Practical bayesian optimization of machine learning algorithms. In *NeurIPS*, 2012. 5

[9] Haoze Sun, Linfeng Jiang, Fan Li, Renjing Pei, Zhixin Wang, Yong Guo, Jiaqi Xu, Haoyu Chen, Jin Han, Fenglong Song, et al. Pocketsr: The super-resolution expert in your pocket mobiles. *arXiv preprint arXiv:2510.03012*, 2025. 9

[10] Jianyi Wang, Zongsheng Yue, Shangchen Zhou, Kelvin CK Chan, and Chen Change Loy. Exploiting diffusion prior for real-world image super-resolution. *International Journal of Computer Vision*, 132(12):5929–5949, 2024. 9

[11] Jingkai Wang, Jue Gong, Lin Zhang, Zheng Chen, Xing Liu, Hong Gu, Yutong Liu, Yulun Zhang, and Xiaokang Yang. Osdface: One-step diffusion model for face restoration. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 12626–12636, 2025. 10

[12] Xintao Wang, Yu Li, Honglun Zhang, and Ying Shan. Towards real-world blind face restoration with generative facial prior. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9168–9178, 2021. 10

[13] Yufei Wang, Wenhan Yang, Xinyuan Chen, Yaohui Wang, Lanqing Guo, Lap-Pui Chau, Ziwei Liu, Yu Qiao, Alex C Kot, and Bihan Wen. Sinsr: diffusion-based image super-resolution in a single step. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 25796–25805, 2024. 9

[14] Pengxu Wei, Ziwei Xie, Hannan Lu, Zongyuan Zhan, Qixiang Ye, Wangmeng Zuo, and Liang Lin. Component divide-and-conquer for real-world image super-resolution. In *European conference on computer vision*, pages 101–117. Springer, 2020. 9

[15] Rongyuan Wu, Lingchen Sun, Zhiyuan Ma, and Lei Zhang. One-step effective diffusion network for real-world image super-resolution. *Advances in Neural Information Processing Systems*, 37:92529–92553, 2024. 9, 10

[16] Rongyuan Wu, Tao Yang, Lingchen Sun, Zhengqiang Zhang, Shuai Li, and Lei Zhang. Seesr: Towards semantics-aware real-world image super-resolution. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 25456–25467, 2024. 9

[17] Peiqing Yang, Shangchen Zhou, Qingyi Tao, and Chen Change Loy. Pgdiff: Guiding diffusion models for versatile face restoration via partial guidance. *Advances in Neural Information Processing Systems*, 36: 32194–32214, 2023. 10

[18] Zongsheng Yue and Chen Change Loy. Difface: Blind face restoration with diffused error contraction. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2024. 10

[19] Zongsheng Yue, Jianyi Wang, and Chen Change Loy. Resshift: Efficient diffusion model for image super-resolution by residual shifting. *Advances in Neural Information Processing Systems*, 36:13294–13307, 2023. 9

[20] Aiping Zhang, Zongsheng Yue, Renjing Pei, Wenqi Ren, and Xiaochun Cao. Degradation-guided one-step image super-resolution with diffusion priors. *arXiv preprint arXiv:2409.17058*, 2024. 9

[21] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 586–595, 2018. 1

[22] Shangchen Zhou, Kelvin Chan, Chongyi Li, and Chen Change Loy. Towards robust blind face restoration with codebook lookup transformer. *Advances in Neural Information Processing Systems*, 35:30599–30611, 2022. 10