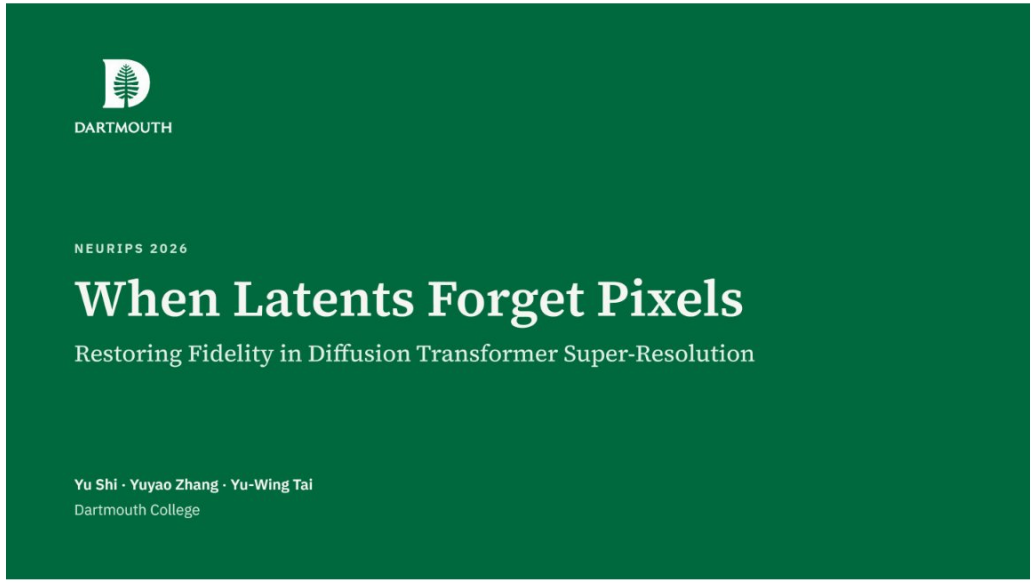
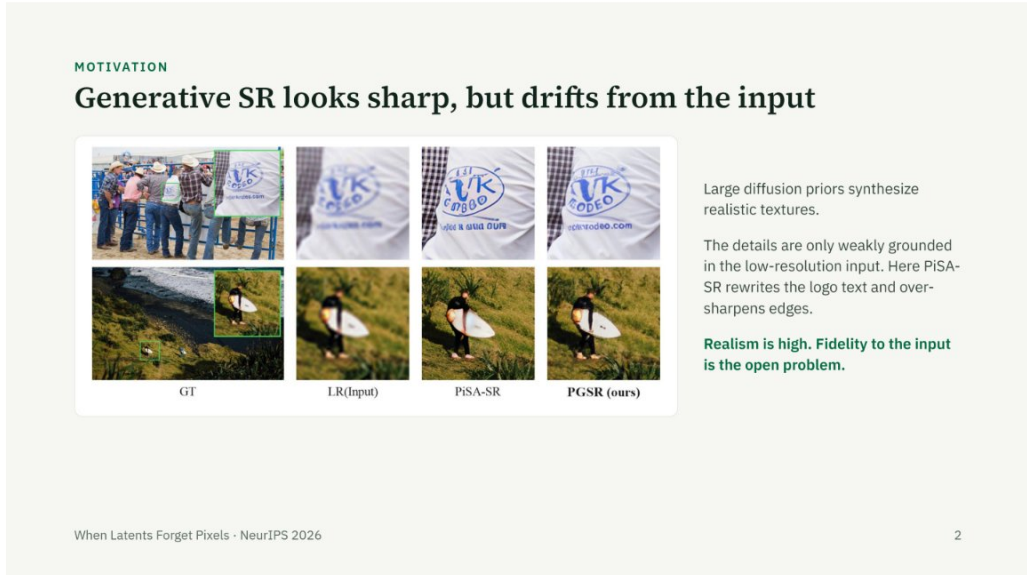


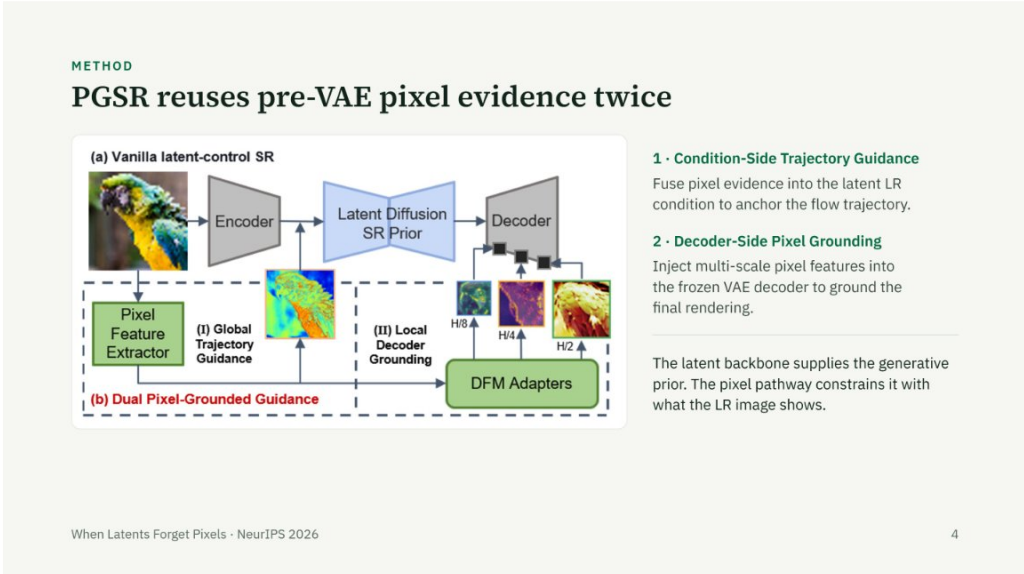
Hi everyone, I’m Yu Shi from Dartmouth College. This is joint work with Yuyao Zhang and Yu-Wing Tai. Our paper is called “When Latents Forget Pixels”. It is about restoring fidelity in super-resolution with diffusion transformers.



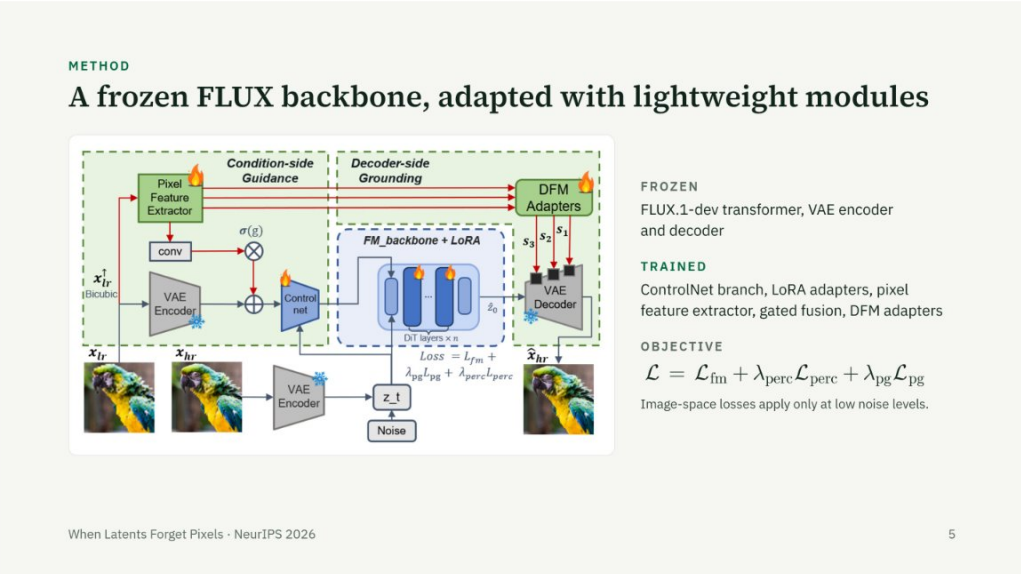
Super-resolution with large generative models now gives very realistic textures. But look at these two examples. The low-resolution input is in the second column. A recent method, PiSA-SR, produces a sharp result, but the text on the shirt is rewritten into something that was never there, and the edges are over-sharpened. The output looks good, but it is no longer faithful to the input. Our result on the right stays much closer to the ground truth. So the open problem is fidelity, not realism.



Why does this happen? Latent diffusion models and diffusion transformers work in a compressed latent space, produced by a VAE. We ran a simple stress test: take a ground-truth image, encode it with the VAE and decode it again. Even with a perfect input, the small numbers and the brand text on this dial become unreadable, for both the Stable Diffusion VAE and the FLUX VAE. So the bottleneck itself is lossy. For super-resolution this leaves two gaps. First, the model is conditioned only on a compressed latent, so the generation trajectory can drift away from the structure in the input. Second, the final image is rendered by a frozen decoder, which can lose more spatial detail.



Our idea is to keep the pixel evidence outside the bottleneck. In a vanilla pipeline, shown at the top, the LR image is seen only after VAE encoding. In our framework, called PGSR, a lightweight pixel feature extractor reads the LR image before compression, and we reuse these features at two places. First, on the condition side, we fuse them into the latent condition, which anchors the generation trajectory. Second, on the decoder side, we inject multi-scale pixel features into the frozen VAE decoder, which grounds the final rendering. The pretrained backbone still provides the generative prior. The pixel pathway constrains it with what the input shows.



Here is the full pipeline. We build on FLUX.1-dev. The transformer backbone and the VAE stay frozen, shown with snowflakes. We train only lightweight parts, shown with flames: a ControlNet branch, LoRA adapters in the backbone, the pixel feature extractor, a gated fusion layer and the decoder adapters. The loss combines the standard flow-matching loss with a perceptual loss and a pixel-grounded L1 loss in image space. The image-space terms are applied only at low noise levels, where the predicted clean latent is reliable. Training has two stages: first on clean bicubic pairs, then with Real-ESRGAN-style degradation.

METHOD

Condition-side guidance anchors the latent trajectory

EXTRACT

A small three-stage CNN reads the upsampled LR image and outputs a 16-channel feature at latent resolution (H/8).

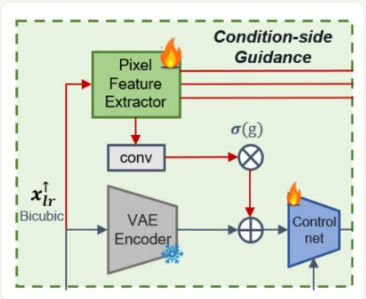
FUSE

$$\tilde{z}_{lr} = z_{lr} + \lambda\,\sigma(g)\,W_p(\bar{p}_{lr})$$

A 1×1 projection, then a residual add to the LR latent, scaled by a learnable gate σ(g). The gate starts open, so pixel evidence is active from the first step.

TRAIN

Trained from scratch, jointly with the ControlNet and LoRA. Gradients come from the flow-matching loss through the ControlNet, and from the image-space losses through the decoder.



The first mechanism is condition-side trajectory guidance. Here is how the feature is produced and fused. A small three-stage convolutional network reads the bicubic-upsampled LR image, before any VAE compression. It outputs a feature map with the same shape as the LR latent: sixteen channels at one-eighth resolution. We project it with a one-by-one convolution and add it to the LR latent as a residual. A learnable scalar gate scales this residual. The gate starts almost fully open, so the pixel evidence is used from the first training step. The fused condition then goes into the ControlNet branch, which steers the frozen transformer. The extractor has no pretraining. It is trained from scratch together with the ControlNet and LoRA. It receives gradients from the flow-matching loss through the ControlNet, and from the image-space losses through the decoder path, which I show next.

METHOD

Decoder-side grounding restores detail at rendering

INJECT

$$\tilde{f}_d^{\ell} = f_d^{\ell} + Z_\ell(\mathrm{SiLU}(A_\ell(s_\ell)))$$

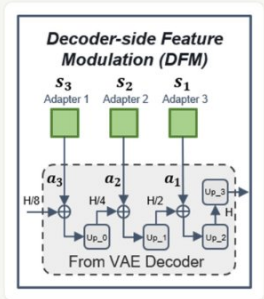
At three decoder stages (H/8, H/4, H/2), the matching pixel feature passes through a 3×3 conv, SiLU and a 1×1 conv, and is added to the decoder activation before the upsampling block.

PRESERVE

The VAE decoder weights stay frozen. The last 1×1 conv is zero-initialized, so training starts from the original decoder and the adapters learn a residual correction.

TRAIN

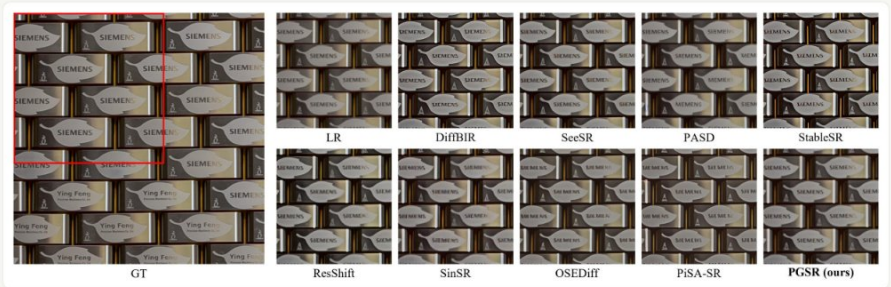
We estimate the clean latent from the predicted velocity, decode it through the DFM decoder, and apply L1 and LPIPS losses against the HR image, only at low noise levels.



The second mechanism works at the decoder. We call it decoder-side feature modulation, or DFM. The pixel extractor also outputs three intermediate feature maps, at one-half, one-quarter and one-eighth resolution. At the three matching stages of the VAE decoder, each feature goes through a small adapter: a three-by-three convolution to match the decoder width, a SiLU, and a one-by-one convolution. The result is added to the decoder activation, right before the upsampling block. It is an additive residual, so the decoder weights stay frozen. The last convolution is zero-initialized, so at the start of training the decoder is exactly the original VAE decoder. To train the adapters, we take the predicted velocity, estimate the clean latent, decode it through this modulated decoder, and compare the image with the ground truth using an L1 loss and LPIPS. We apply these image-space losses only at low noise levels, where the estimated latent is reliable. The same losses also update the pixel extractor.

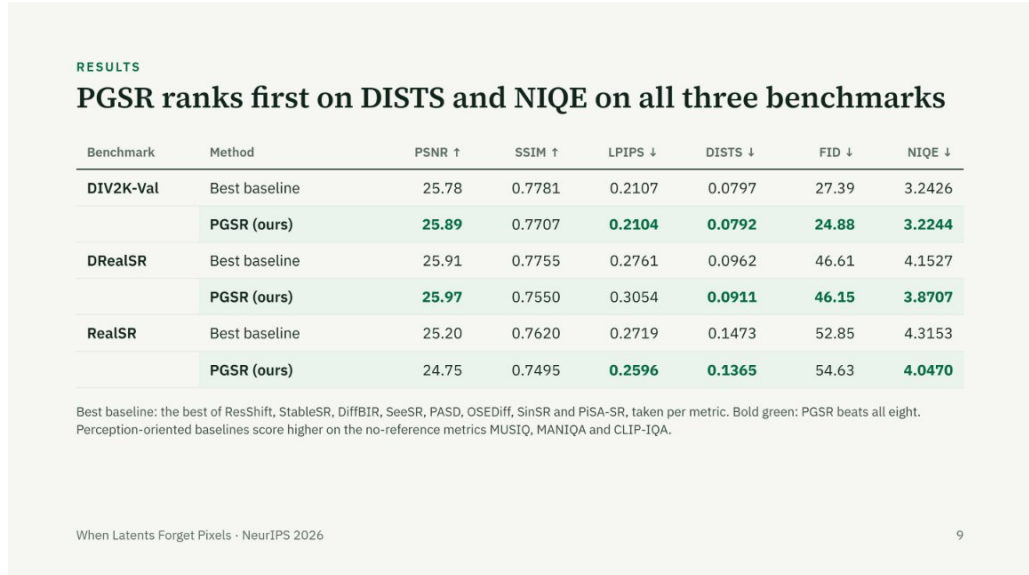
RESULTS

On real captures, PGSR keeps structure and text faithful



RealSR capture, ×4. Several baselines hallucinate textures or distort the lettering. PGSR preserves object boundaries and renders the text correctly.

Now the results. This is a real captured image from the RealSR benchmark, with real degradation rather than synthetic downsampling. The red box in the ground truth marks the zoomed region. Look at the lettering on the nameplates. Several baselines hallucinate textures or distort the letters into something unreadable. PGSR keeps the object boundaries and renders the text correctly, because the pixel evidence from the input constrains both generation and decoding.



For the numbers, we compare with eight generative SR methods on three ×4 benchmarks: DIV2K validation with bicubic degradation, and the real-world sets DRealSR and RealSR. For each metric, this table shows the best of the eight baselines next to PGSR. Bold green means PGSR is better than all eight. On DIV2K, PGSR is best on PSNR, LPIPS, DISTS, FID and NIQE. On DRealSR, it is best on PSNR, DISTS, FID and NIQE. On RealSR, it is best on LPIPS, DISTS and NIQE. To be fair, perception-oriented baselines still score higher on some no-reference metrics such as MUSIQ, MANIQA and CLIP-IQA. Overall, PGSR gives a better balance between realism and fidelity.

RESULTS

Both pathways help, and more small text is preserved



Variant	DIV2K PSNR ↑	DIV2K LPIPS ↓	RealSR PSNR ↑	RealSR LPIPS ↓
w/o pixel grounding	23.68	0.2431	22.15	0.2789
w/o condition-side	24.66	0.2204	24.22	0.2631
w/o decoder-side	24.78	0.2188	24.08	0.2626
PGSR (full)	**25.89**	**0.2104**	**24.75**	**0.2596**

SMALL-TEXT OCR

1,339 words, 4–8 LR pixels high, from 100 TextOCR-val images at bicubic ×4. Recognizer: PARSeq.

	HR ref.	PASD	SeeSR	PiSA-SR	PGSR
Word acc. ↑	86.41%	8.36%	7.17%	7.32%	**27.48%**
CER ↓	4.97%	82.00%	85.77%	82.40%	**57.33%**

The ablation shows that both pathways matter. Without any pixel grounding, PSNR on DIV2K drops from 25.89 to 23.68, and the text on this ATM sign is garbled. Removing only the condition-side guidance or only the decoder-side grounding also hurts, on both benchmarks. The full model is the best, so the two mechanisms are complementary. Below is a text-preservation test. We took 1,339 small words from the TextOCR validation set. In the LR image they are only 4 to 8 pixels high. After ×4 super-resolution, we ran a text recognizer, PARSeq, on the same ground-truth boxes. On the HR images the recognizer reaches 86 percent word accuracy, which is the upper bound. With PASD, SeeSR and PiSA-SR, accuracy is around 7 to 8 percent. PGSR reaches 27.5 percent, and the character error rate drops from above 80 percent to 57 percent. There is still a large gap to the HR reference, but much more of the original text survives.

RESULTS

Sparse attention cuts inference time and scales to 8K



2K → 8K super-resolution on DIV8K, using sparse attention. Zoom: LR input (top) and PGSR output (bottom).

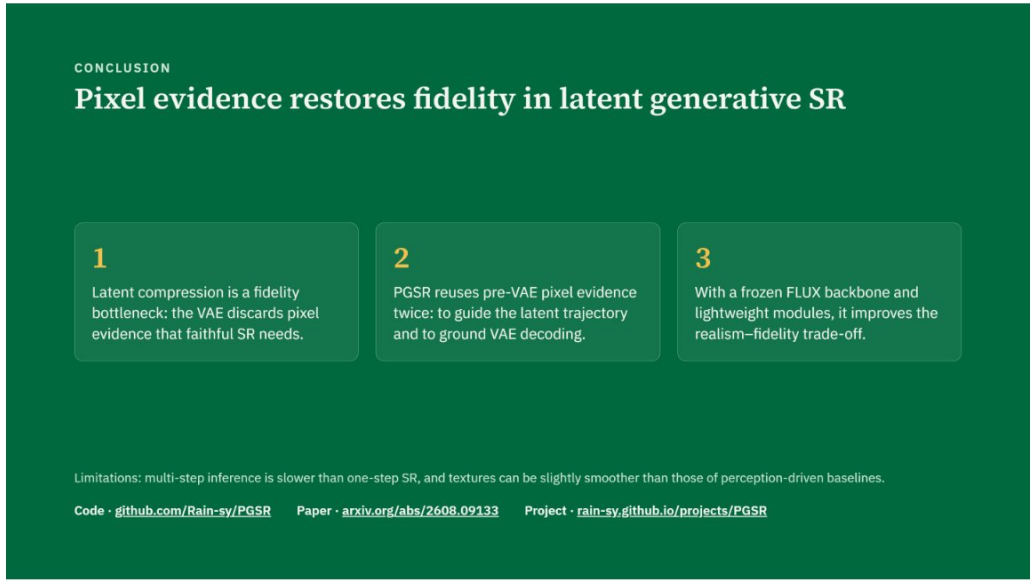
SPARSE ATTENTION

Local-window attention replaces dense image-token attention in selected blocks. Inference takes about 37% less time, for a modest quality drop.

Benchmark	Attention	s/img ↓	PSNR ↑	LPIPS ↓
DIV2K, 1792²	Full	61.81	26.36	0.2571
	Sparse	**38.95**	25.81	0.2603
RealSR, 1536²	Full	58.28	24.88	0.2582
	Sparse	**37.09**	24.37	0.2601

Seconds per image at a fixed output size, 20 sampling steps.

This slide is about efficiency. FLUX is a large model, and dense attention over all image tokens is expensive at high resolution. So we also trained a variant with sparse, local-window attention in selected blocks, using the same pixel-grounding pathway. At a fixed output size of 1792 by 1792 on DIV2K, the time per image drops from about 62 seconds to 39 seconds with 20 sampling steps. That is about 37 percent less. On RealSR the saving is similar. The cost is a modest drop in quality: about half a dB in PSNR, and almost no change in LPIPS. Sparse attention also makes very large outputs practical. On the left is a 2K to 8K example from the DIV8K dataset. In the zoomed crop, the carved relief on the monument is restored cleanly from a very blocky input.



To conclude. First, latent compression is a real fidelity bottleneck for generative super-resolution: the VAE discards pixel evidence that faithful restoration needs. Second, PGSR keeps that evidence outside the bottleneck and reuses it twice, to guide the latent trajectory and to ground the VAE decoding. Third, this works as a lightweight adaptation of a frozen FLUX backbone, and it improves the trade-off between realism and fidelity. There are limitations: inference is multi-step, so it is slower than one-step models, and textures can be slightly smoother than those of perception-driven methods. The code is available on GitHub. Thank you for listening.