A Supplementary Materials

A.1 Code Availability

The implementation of our methods along with the datasets used will be made publicly available.

A.2 Comparison to Self-Validation

Self-validation, which uses a subset of measurements for validation, is popular in single-instance MRI reconstruction for preventing overfitting (Yaman et al., 2021; Darestani et al., 2022). It works by detecting the timing for stopping as near to the peak PSNR as possible. Our method differs in that it can generally enhance the peak PSNR of the architecture while also alleviating overfitting. This can be seen from the results of $\mathbf{A_{2\text{-}256}}$, $\mathbf{A_{2\text{-}64}}$, and Fig. 7 (main text). We show below that they can be combined to achieve much better performance.

Table 1: Quantitative evaluation on $4\times$ multi-coil **knee** datasets. The best and the second-best are highlighted. 5% of the measurements are held out for validation. 'ws' denotes the duration (# iterations) of a sliding window that monitors the self-validation error for **automatic early stopping**.

Methods	$\mathbf{A_{2\text{-}256}}$	$\mathbf{A_{2\text{-}64}}$	$\mathbf{A_{5\text{-}256}}$	$\mathbf{A_{5\text{-}64}}$	$\mathbf{A_{2\text{-}256}}$	$\mathbf{A_{2\text{-}64}}$	$\mathbf{A_{5\text{-}256}}$	$\mathbf{A_{5\text{-}64}}$
	PSNR (↑)				SSIM (↑)			
Self-Val. (ws=30)	29.59	29.59	31.18	31.05	0.682	0.695	0.746	0.744
Self-Val. (ws=50)	29.04	29.62	31.07	30.94	0.642	0.684	0.738	0.737
Ours (3000 iters)	31.61	31.93	29.40	31.67	0.750	0.776	0.702	0.727
Ours w. Self-Val (ws=30)	31.49	31.09	31.74	31.73	0.762	0.762	0.769	0.772
Ours. w. Self-Val (ws=50)	31.60	31.41	31.78	31.63	0.762	0.767	0.771	0.771
Baseline (3000 iters)	27.18	27.62	29.16	29.23	0.541	0.575	0.625	0.640

Table 2: Quantitative evaluation on $4\times$ multi-coil **brain** datasets.

Methods	$\mathbf{A_{2\text{-}256}}$	$\mathbf{A_{2\text{-}64}}$	$\mathbf{A_{5\text{-}256}}$	$\mathbf{A_{5\text{-}64}}$	$\mathbf{A_{2\text{-}256}}$	$\mathbf{A_{2\text{-}64}}$	$\mathbf{A_{5\text{-}256}}$	$\mathbf{A_{5\text{-}64}}$
	PSNR ↑				SSIM ↑			
Self-Val. (ws=30)	30.39	30.06	32.78	32.48	0.822	0.832	0.872	0.868
Self-Val. (ws=50)	30.21	30.15	32.77	32.44	0.813	0.829	0.870	0.867
Ours (3000 iters)	32.90	33.12	32.08	32.83	0.855	0.870	0.815	0.851
Ours w. Self-Val (ws=30)	32.94	32.56	33.06	33.04	0.874	0.873	0.880	0.879
Ours. w. Self-Val (ws=50)	32.99	32.72	33.06	32.52	0.874	0.874	0.880	0.870
Baseline (3000 iters)	29.08	29.41	31.15	31.42	0.729	0.761	0.782	0.801

A.3 Comparisons to supervised methods

Supervised methods shine when test data are within the training distribution. DIP-like methods are more advantageous on out-of-distribution data as they are agnostic to changes in acquisition protocols and anatomy shift, etc.(Yaman et al., 2021). Our method accelerates DIP by allowing a more compact network to be employed, and when combined with self-validation, its runtime is further reduced.

Table 3: Robustness and runtime comparisons with U-Net on the $4\times$ multi-coil **brain** validation dataset. **In-domain** dataset: 50 AXT1PRE slices. **Out-domain** dataset: 30 AXFLAIR slices. Runtime is computed as the per-slice average for every slice of size $20 \times 640 \times 320$. The DIP $\mathbf{A_{2\text{-}64}}$ is trained for 3000 iterations when self-validation is not used.

	Methods	In-domain		Out-domain		Runtime (mean±std)	
		PSNR	SSIM	PSNR	SSIM	Train	Inference
Trained	U-Net	**34.11**	**0.910**	28.25	0.785	$\geq$ 3 days	0.1 ± 0.003 sec
Untrained	$\mathbf{A_{2\text{-}64}}$ (baseline)	29.41	0.761	29.77	0.715	–	26.5 ± 8.1 mins
	$\mathbf{A_{2\text{-}64}}$ (ours)	33.12	0.870	**32.45**	0.832	–	26.8 ± 8.3 mins
	$\mathbf{A_{2\text{-}64}}$ (ours) w. Self-Val.	32.56	0.873	32.11	**0.840**	–	4.8 ± 2.7 mins

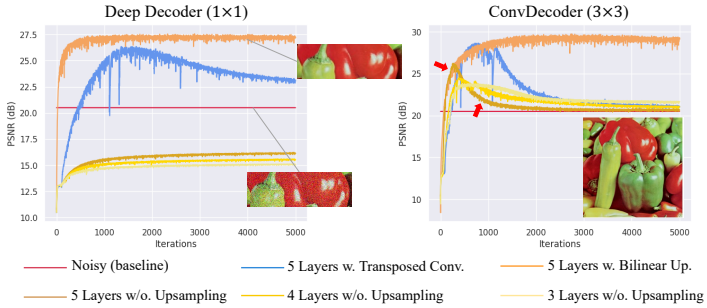

Figure 8: **Denoising** experiments. (**Left**) In non-convolutional networks, removing the upsampling hampers the denoising capability, which cannot be compensated by merely adjusting the network to be more under-parameterized. Transposed convolutions result in a more rapid decline in performance than bilinear upsampling. (**Right**) Convolutional layers *alone* exhibit certain denoising effects but necessitate early stopping. The showcased image is from the classic dataset Set9 (Dabov et al., 2007).

A.4 THE "DEVIL" IS IN THE UPSAMPLING

Here we provide additional evidence demonstrated on brain datasets as well as natural images to support our findings about the unlearnt upsampling and its relationships with other architectural properties in DIP. These findings critically motivate our methods and lead us to conclude that the underperformance in DIP is not primarily attributed to the number of parameters.

A.4.1 ADDITIONAL MRI EXPERIMENTS

As stated in Sec.4, an *unlearnt* upsampler can be seen as a zero insertion step which increases the output sampling rate, followed by a non-ideal low-pass filter (LPF, shortened as $\mathcal{L}$) that attenuates both the introduced high-frequency replica and signals. Bilinear and nearest neighbor (NN) upsamplers differ only in the LPFs used. We additionally constructed a customized upsampler that has a greater attenuation ability than bilinear upsampling. This was done by first interleaving the feature maps of every layer with zeros and then convolving them with a handcrafted LPF: $\mathcal{L}_{-100}$, with the subscript denoting the decayed dB. The Details of construction are specified in A.7.

Table 4: **Importance of upsampling**. Evaluated on the $4\times$ multi-coil **brain** dataset. From the left to the right, the attenuation extent of the LPF increases. PSNR values at 3000th iteration are reported.

Methods	w/o. Upsampling.	NN	Bilinear	$\mathcal{L}_{-100}$	# of Params. (Millions)
ConvDecoder	28.69 ± 1.6	31.78 ± 1.2	32.31 ± 1.3	32.48 ± 1.2	4.1 M
Deep Decoder	24.55 ± 1.1	27.10 ± 0.9	31.36 ± 1.4	32.68 ± 1.1	0.47 M

Tab. 4 shows that simply varying the upsampling type substantially influences the network performance such that the performance gap between the two networks can even be closed without requiring architecture scaling. Overall, the presence of unlearnt upsampling is vital to the non-convolutional Deep Decoder and enhances both the accuracy and stability of ConvDecoder: the peak PSNR is reached more slowly when the attenuation is stronger, alleviating overfitting (Fig. 9).

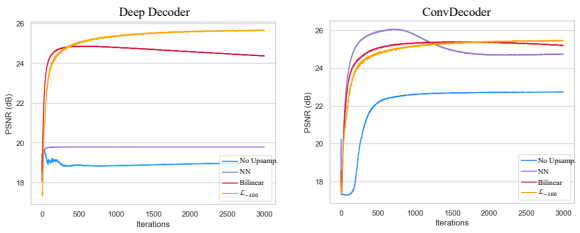

Figure 9: Results evaluated on the masked regions averaged across 30 slices. The unlearnt upsampler critically influences both the peak PSNR and the susceptibility to overfitting.

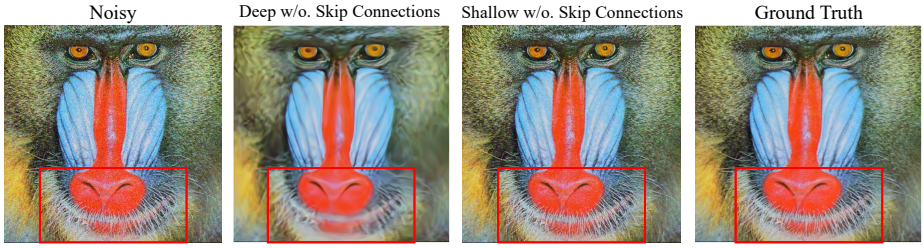

Figure 10: **Denoising** experiments. Deeper architectures with few or no skip connections tend to generate smoother outputs compared to the shallower ones.

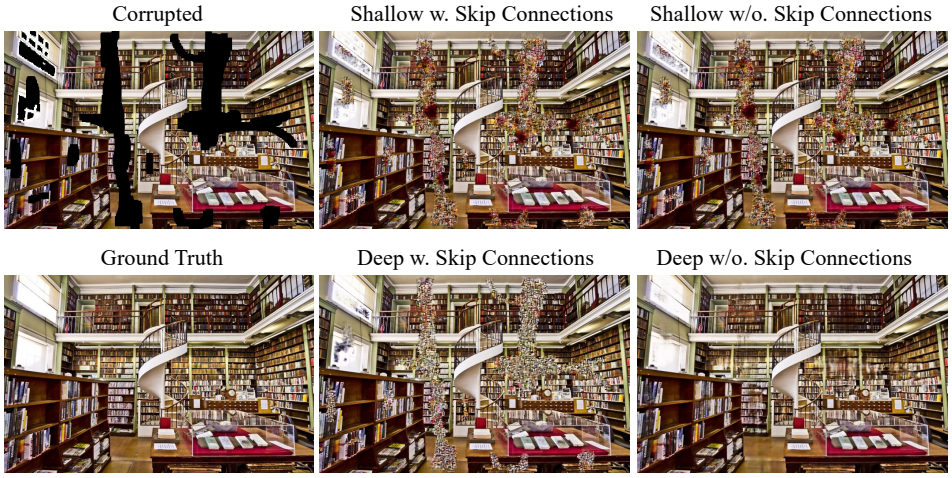

Figure 11: **Inpainting** experiments. Deeper architectures with few or no skip connections tend to generate smoother predictions for the masked regions than the shallower architectures. Skip connections make deep architectures perform similarly as the shallower ones.

A.4.2 Natural Image Experiments

We reaffirmed our observations above on **image denoising**, which is a natural application of DIP. The results in Fig. 8 show a very similar trend as in MRI reconstruction. We further validated on a challenging **image inpainting task** that inherently resembles the case in MRI reconstruction. The results are shown in Fig. 11 and Fig. 12

We argue that the understanding about the upsampling and its interactions with other architecture elements can help explain why deeper networks with fewer skip connections converge more slowly, generate smoother outputs and are less prone to overfitting (Sec.4 in main text). Concretely, the upsampling operation inserted in-between the decoder layer slows down the generation of high frequencies required for transforming the lower-resolution feature maps into the higher-resolution target image, primarily due to its role as a fixed low-pass filter. As the network depth increases, the degree of smoothness increases (Fig. 10). Skip connections notably accelerate the convergence (Fig. 12) and ameliorate the over-smoothing issue, likely due to the reduced "effective" upsampling rate. All these observations are consistent with our MRI experiments in Sec.4 (main text).

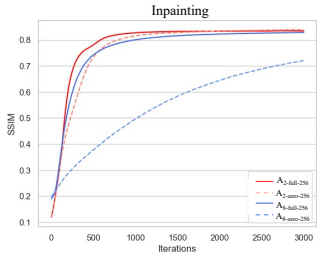

Figure 12: Deep architectures with zero skip connection converge more slowly, i.e., $\mathbf{A}_{\mathbf{8\text{-}zero\text{-}256}}$

A.5 EXAMPLE RESULTS ON 8× UNDERSAMPLING

DIP with an inappropriately chosen architecture exhibits even more severe reconstruction artifacts in 8× undersampling, which may not be remedied by early stopping as even the peak PSNR could be low (see metric curves). Nevertheless, our method substantially alleviates the artifacts while employing the same architecture. Particularly, we found that scaling up not only the sigma but also the kernel size of the Gaussian blur improves the visual quality in such a high undersampling rate.

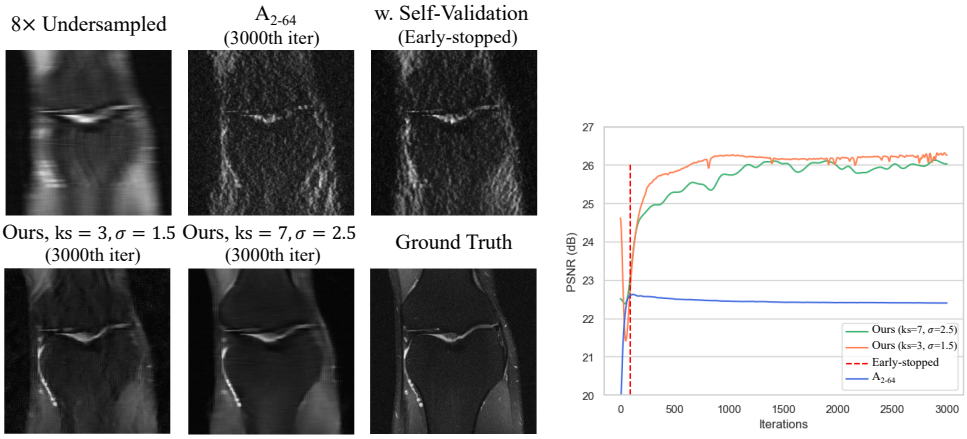

Figure 13: Qualitative results of 8× undersampling. All methods were evaluated on $\mathbf{A_{2\text{-}64}}$.

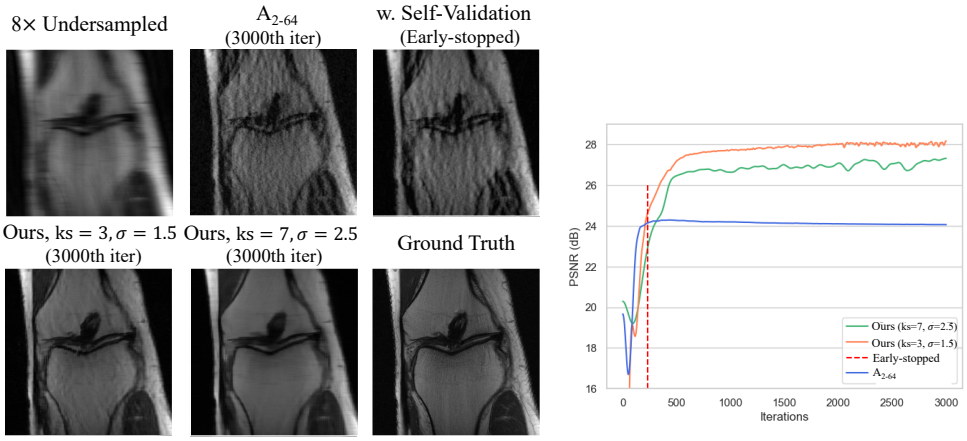

Figure 14: Qualitative results of 8× undersampling. All methods were evaluated on $\mathbf{A_{2\text{-}64}}$.

A.6 ANALYSIS ON SENSITIVITY TO HYPERPARAMETERS

As stated in the "implementation details" section, we set the filter size of the Gaussian blur to a fixed value, i.e., 3, and chose the sigma uniformly from a fixed range, i.e., $[0.5, 2.0]$. The substantially improved performance demonstrates that the method exhibits robustness to a certain range of hyperparameters. For the undersampling rate higher than 4×, a larger kernel size and sigma value are generally beneficial for better visual quality (see qualitative examples in Sec. A.5). We then test the sensitivity of the proposed Lipschitz regularization to its only hyperparameter - the regularization coefficient λ. The experiments were performed on the multi-coil knee validation dataset (Tab. 5).

A.7 DETAILS OF THE CUSTOMIZED UPSAMPLER

The upsamplers experimented in Tab. 4 is constructed by first inserting zeros into the input (or feature maps) in an interleaved fashion, and then convolving with the filter with the following coefficients:

Algorithm 1: PyTorch-style pseudocode for customized upsampling

```
# upx:  the upsampling scaling factor in the x direction
# upy:  the upsampling scaling factor in the y direction
# x:  the input to be upsampled
def InsertZeros(x, upx, upy, gain=1.0):
   b,c,h,w = x.size()
   x = x.reshape([b, c, h, 1, w, 1])
   x = F.pad(x, [0, upx - 1, 0, 0, 0, upy - 1])
   x = x.reshape([b, c, h * upx, w * upy])
   x = x * gain
   return x
# LPF construction
# w:  the coefficients
def lowpass_conv(num_chns, w, pad_size='same', pad_mode='zeros'):
   # filter size
   k_size = len(w)
   # Convert 1D LPF coefficients to 2D
   f_2d_coeff = torch.outer(w,w)
   f_weights = torch.broadcast_to(f_2d_coeff, [num_chns, 1, k_size,
    k_size])
   conv = nn.Conv2d(num_chns, num_chns, kernel_size=k_size,
    stride=1, padding=pad_size, padding_mode=pad_mode, bias=False,
    groups=num_chns)
   conv.weight.data = f_weights
   conv.weight.requires_grad = False
   return conv
```

Table 5: **Evaluation on hyperparameter sensitivity** of the Lipschitz regularization. PSNR values (↑) are reported. The chosen is underlined.

Matrix norm	Hyper-param.	$\mathbf{A}_{2\text{-}256}$	$\mathbf{A}_{2\text{-}64}$	$\mathbf{A}_{5\text{-}256}$	$\mathbf{A}_{5\text{-}64}$	$\mathbf{A}_{8\text{-}256}$	$\mathbf{A}_{8\text{-}64}$
ℓ_∞	$\underline{\lambda = 1}$	28.41	29.21	29.17	29.79	29.43	30.14
	$\lambda = 1.5$	27.89	28.98	28.68	30.11	29.13	29.42
	$\lambda = 2$	28.36	29.25	28.51	29.60	28.98	29.52

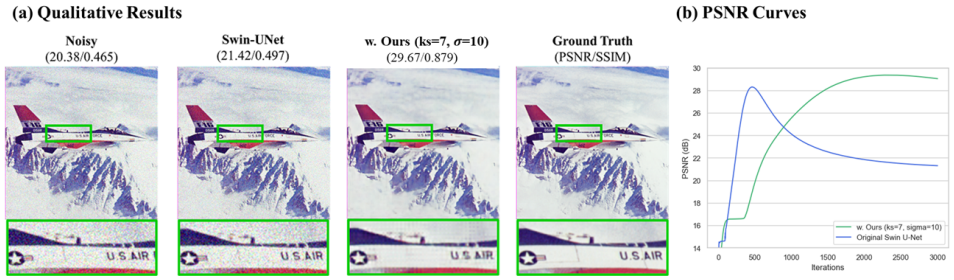

Figure 15: Example results of a transformer (i.e., Swin U-Net Cao et al. (2022)). The original Swin U-Net consists of only Swin Transformer blocks and skip connections, without upsampling layers. Our method substantially alleviates the overfitting and enhances the peak PSNR.

Nearest neighbor (NN): $[0.5, 0.5]$

Bilinear: $[0.25, 0.5, 0.25]$

$\mathcal{L}_{-100}$: $[0.000015, 0.000541, 0.003707, 0.014130, 0.037396, 0.075367, 0.121291,$
$0.159962, 0.175182, 0.159962, 0.121291, 0.075367, 0.037396, 0.014130, 0.003707, 0.000541, 0.000015]$

$\mathcal{L}_{-100}$ is designed using the Kaiser window, with the cutoff frequency as 0.1 and the Beta of the Kaiser window as 10.

Specifically, the customized filter can be constructed using the following code and can then be used as a plug-in module for any network architecture.

REFERENCES

Hu Cao, Yueyue Wang, Joy Chen, Dongsheng Jiang, Xiaopeng Zhang, Qi Tian, and Manning Wang. Swin-unet: Unet-like pure transformer for medical image segmentation. In *European conference on computer vision*, pp. 205–218. Springer, 2022.

Kostadin Dabov, Alessandro Foi, and Karen Egiazarian. Video denoising by sparse 3d transform-domain collaborative filtering [c]. In *Proc. 15th European Signal Processing Conference*, volume 1, pp. 7, 2007.

Mohammad Zalbagi Darestani, Jiayu Liu, and Reinhard Heckel. Test-time training can close the natural distribution shift performance gap in deep learning based compressed sensing. In *International Conference on Machine Learning*, pp. 4754–4776. PMLR, 2022.

Burhaneddin Yaman, Seyed Amir Hossein Hosseini, and Mehmet Akçakaya. Zero-shot self-supervised learning for mri reconstruction. *arXiv preprint arXiv:2102.07737*, 2021.