In Search of a Data Transformation That Accelerates Neural Field Training

Supplementary Material

Contents

Architecture	Dataset	Original	Random pixel perm.	Zigzag perm.	Inversion	Standardization	Linear Scaling		Centering		Gamma Correction	
							$t = 0.5$	$t = 2.0$	$t = 1.0$	$t = 2.0$	$\gamma = 0.5$	$\gamma = 2.0$
SIREN	Kodak	1371.4	1100.5	79.8	2271.4	706.7	1875.2	1051.0	2071.7	1591.7	4297.1	1835.4
	DIV2K	1568.6	1366.3	73.3	2512.1	778.8	2033.6	1153.6	1892.2	1532.3	6630.2	2102.6
	CLIC	1257.1	1233.7	73.5	2089.9	683.6	1832.4	1014.3	1663.3	1334.1	6359.3	1806.8
Instant-NGP	Kodak	207.3	217.1	240.6	3121.9	761.7	168.3	304.0	167.1	258.6	1861.2	677.3
	DIV2K	194.0	146.7	247.1	4240.3	439.1	164.6	505.6	151.4	245.2	4207.9	1369.1
	CLIC	199.0	151.4	238.5	3881.8	490.4	168.2	502.3	152.0	257.4	4773.7	1418.4

Table S4. **Average steps for achieving PSNR 50dB.** We report the average steps of 8 data transformations (including original) under six different combinations of datasets and models. The instances that do not achieve the target PSNR with any of the learning rates are excluded from the average calculation. Gamma correction with γ = 2.0 fails to fit (1, 2, 1) images on Kodak, DIV2K, and CLIC datasets, respectively. Similarly, γ = 0.5 and Linear Scaling with t = 2.0 do not fit (0, 6, 2) and (1, 2, 1) images in these datasets, respectively.

S1. Wall-clock latency comparison

We primarily focus on evaluating the speedup through our proposed *acceleration factor*, which is derived from the number of SGD steps. In Tab. S5, we show whether this factor indeed leads to an acceleration of neural field training. To demonstrate this point empirically, we conduct a comparison between the average wall-clock runtimes of two key stages: data pre-processing and loading (referred to as "Load") and the actual SGD iterations (referred to as "Train") on original and RPP Kodak images. We find that the extra computation spent during the "Load" is much smaller in scale than the speedup from "Train", resulting in a net speedup. This corresponds to a time savings of 54543 ms, a whopping 25% time reduction, in the SIREN architecture, where SGD iterations make up the majority of the training.

	SIREN		Instant-NGP	
	Load (ms)	Train (ms)	Load (ms)	Train (ms)
Original	2,913	212,170	1,133	8,484
RPP	2,933	157,607	1,324	7,813
	(+20)	(-54,563)	(+191)	(-671)
Speedup		54,543		480

Table S5. **Wall-clock speedup (ms) on Kodak dataset.**
Note. The net speedup may depend on the choice of hardware; we measure the wall-clock speed on GPU server with a NVIDIA GeForce RTX 3090, and AMD EPYC 7313 16-Core CPU. The speedup can be even larger in CPU-only setups where training is much slower (e.g., mobile devices).

S2. Average number of steps for achieving target PSNR

In Tab. S4, we present the average number of steps required to achieve a 50dB Peak Signal-to-Noise Ratio (PSNR). We observe that this table does not exactly reflect the trends observed in Tab. 1 of the main paper. This discrepancy is due to the presence of *outliers*; some images that require much bigger number of steps than other images can dominate the overall statistic. For example, for Instant-NGP, the original Kodak#20 image requires 1279 number of steps until convergence, which is 7.96 times greater than other images on average. In such case, the average depends heavily on this sample; without Kodak#20, the average for original is 160.7 steps and for RPP is 111.0 steps. The acceleration factor, which we used for the main table, is relatively robust against this issue.

Note. For some data augmentations, several augmented images could not have been fit with Instant-NGP to PSNR 50dB, with any of the learning rates that we tried. In particular, Gamma correction with $\gamma = 2.0$ cannot fit $(1, 2, 1)$ images on Kodak, DIV2K, and CLIC datasets, respectively. Likewise, Gamma correction with $\gamma = 0.5$ and Linear Scaling with $t = 2.0$ cannot fit $(0, 6, 2)$ and $(1, 2, 1)$ images in each datasets, respectively. For these cases, we report the average number of steps *except* on unconverged cases. Nevertheless, the average number of steps for PSNR 50dB on other data points for these augmentations are typically very large; thus it is very unlikely that these modified averages will lead to a misleading conclusion that such augmentations are beneficial for the training speed.

S3. Extended discussions on the potentials and limitations of data transformations

While RPP provides consistent speedups for fitting the given datum, it has limited applicability to scenarios that require the trained model to have certain characteristics, such as interpolatability (as briefly discussed Sec. 2.2). This section provides a more comprehensive discussion on this matter, to elucidate both the potential benefits and limitations.

Image fitting is to represent the target 2D image with a corresponding neural field [24, 34]. Here, the main challenge is to *overfit* to a singular data instance with high fidelity, and our framework is well-suited for this application. By applying the same permutation matrix across all images, RPP can accelerate training without additional memory overhead (Appendix S4).

Image superresolution [4, 14, 47] and **image inpainting** [30, 41] utilizes neural field trained on a set of seen coordinates to predict the signal value on unseen coordinates. These applications strongly rely on the ability of neural field to capture the spatial patterns of the target signal from the seen coordinates and interpolate them. However, RPP introduces a significant challenge in learning the meaningful implicit information in terms of spatial relationships. RPP can disrupt the neural field's understanding of locality and continuity, which is vital for both superresolution and inpainting. Other transformations that adjust only pixel intensities (*e.g.*, standardization), retain the local structure of data, and can thus be used.

Data compression [6, 8, 17] reduces the required number of bits to store the data by representing the datum by a neural field with a small number of parameters. This idea, however, entails substantial computational resources for each datum during the encoding process, where we need to train a neural field. RPP is potentially very useful for reducing the encoding time, by accelerating the neural field training.

3D scene reconstruction and novel view synthesis [25, 26] utilizes a neural field trained on a small number of 2D images of a same 3D scene to generate a novel view of the scene. For this, we heavily rely on the capability of the neural field to infer implicit relationships from all input data. Data transformations involving pixel relocations present a significant hurdle in this context, complicating or outright precluding the networks to predict the adequate relationship. Several concurrent works [20, 22] have explored a training paradigm which trains a single input with segmenting based on the inherent characteristics such as frequencies. Our hypothesis, "blessing of no pattern", can provide a novel insight within this discourse. We demonstrate that there exist representative patterns in data, and they work completely differently depending on the learning phase. From this point of view, the representative patterns can serve as a new criterion for partitioning an instance.

Inference with neural field weights [5, 7] regards the set of weight parameters derived from neural fields as a conventional input feature, based on which we perform inference. Data transformations can help constructing a large-scale dataset of neural field weights, which can be used to train a model that predicts based on the weight vectors.

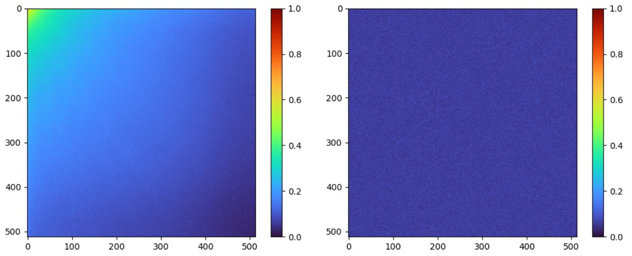
S4. Utilizing a single permutation matrix for RPP

In the main paper, we used an independently drawn permutation matrix for each RPP image. Despite the interesting results, this hinders applying RPP to further applications due to its complete randomness and memory footprint of the matrix.

	Independently drawn	Same permutation
SIREN	1.26×	1.27×
Instant-NGP	1.50×	1.51×

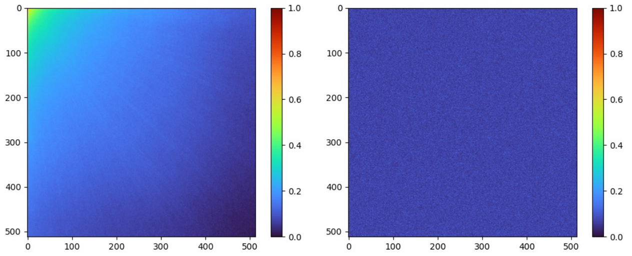
Table S6. Using independently drawn vs. same permutation.

We find that RPP can accelerate the neural field training even with *a single unified permutation matrix*. In Tab. S6, we compare the acceleration factors of RPP using the independently drawn random permutation (per image) and using the same permutation, on the Kodak dataset. We observe that the quantities are roughly identical to each other both in two architectures.



(a) Original (b) Permutation

Figure S9. **Frequency spectra of original vs. RPP (DIV2K).** We compare the average DCT coefficients of the original and RPP DIV2K images. Upper left region denotes the low-frequency, and the lower right region denotes the high-frequency.



(a) Original (b) Permutation

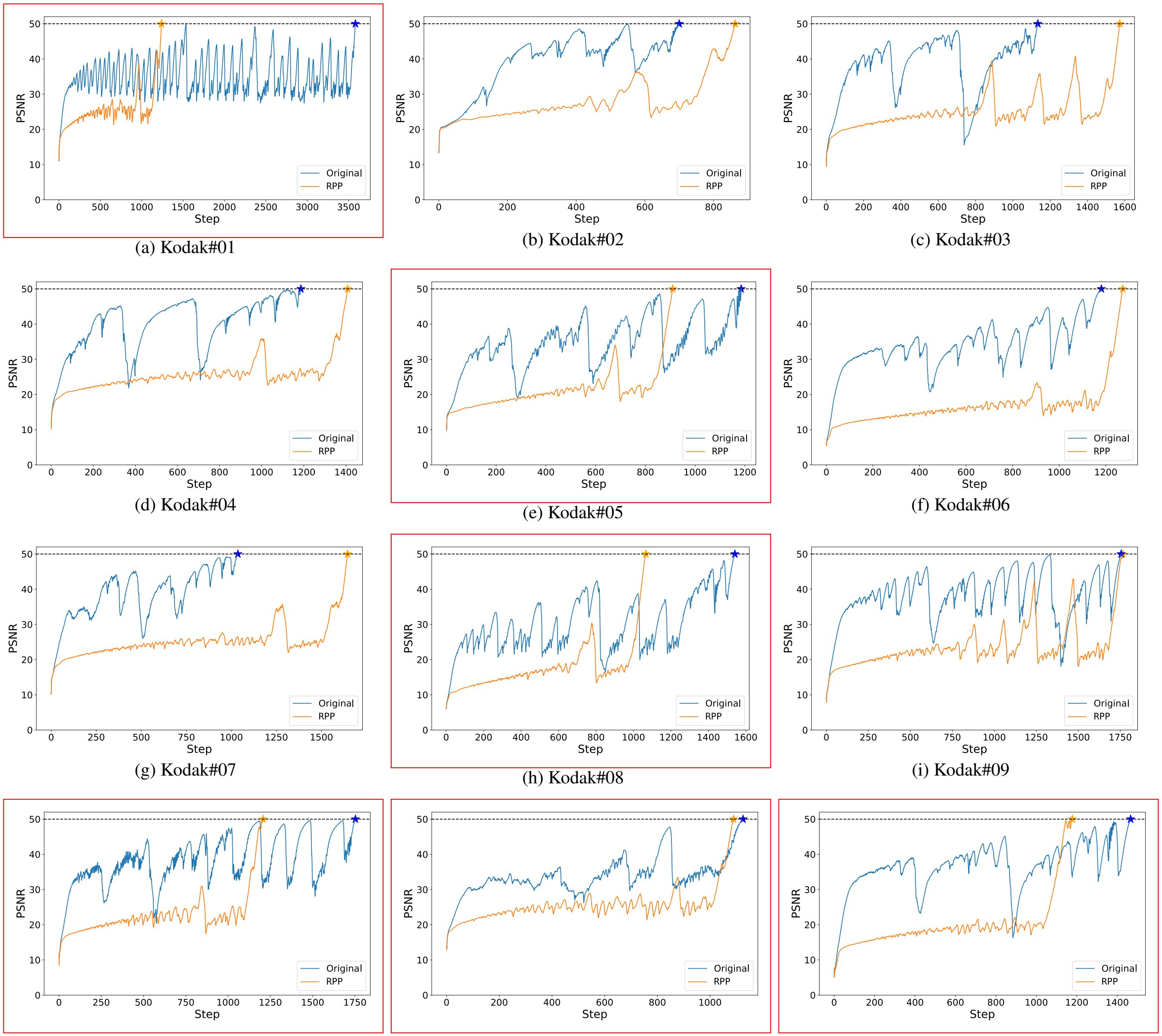
Figure S10. **Frequency spectra of original vs. RPP (CLIC).** We compare the average DCT coefficients of the original and RPP CLIC images.

S5. DCT coefficient of other datasets

In Fig. 3 of the main text, we have compared the average Discrete Cosine Transform (DCT) coefficients of original images and RPP images on the Kodak dataset. Figure 3 demonstrates a discernible pattern: in the original images, we observe that the upper left region (low-frequency components) has much higher coefficients than other regions, whereas in RPP images the scale of the coefficients are relatively uniform. We extend our analysis to other datasets, DIV2K and CLIC, to validate the consistency of our initial findings. The results from these datasets (Figs. S9 and S10, respectively) mirror the observations on the Kodak images. In the original images from DIV2K and CLIC, we once again observed a prevalence of low-frequency components, particularly in the upper left region of the frequency spectra. The distribution is uniform on RPP images.

S6. Full PSNR curves on the Kodak dataset

Figures S11 and S12 show the PSNR curves of a total of 24 images in Kodak dataset. RPP images fit faster than original images on 15 out of 24 images (marked with red borders). Throughout all images, the RPP images quickly surge to PSNR 50dB at a later stage in the majority of cases. In contrast, the original images show fluctuating PSNR values, which mostly hover above a moderate level, *i.e.* 30dB.



(a) Kodak#01

(b) Kodak#02

(c) Kodak#03

(d) Kodak#04

(e) Kodak#05

(f) Kodak#06

(g) Kodak#07

(h) Kodak#08

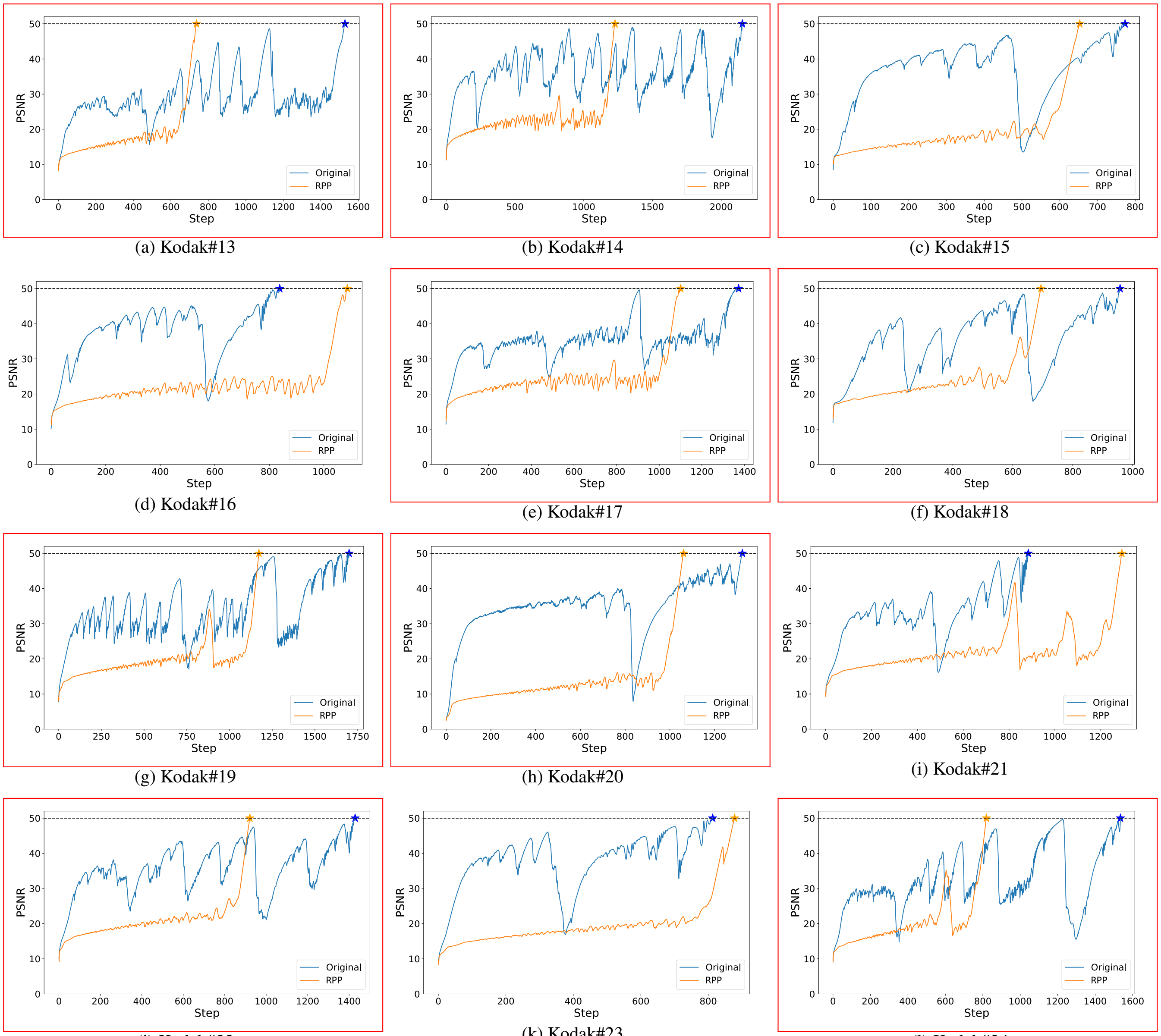
(i) Kodak#09

(j) Kodak#10

(k) Kodak#11

(l) Kodak#12

Figure S11. **PSNR Curves of Kodak images #01–#12.** We report the PSNR curves of Kodak images. The RPP images quickly surge to PSNR 50dB.



(a) Kodak#13

(b) Kodak#14

(c) Kodak#15

(d) Kodak#16

(e) Kodak#17

(f) Kodak#18

(g) Kodak#19

(h) Kodak#20

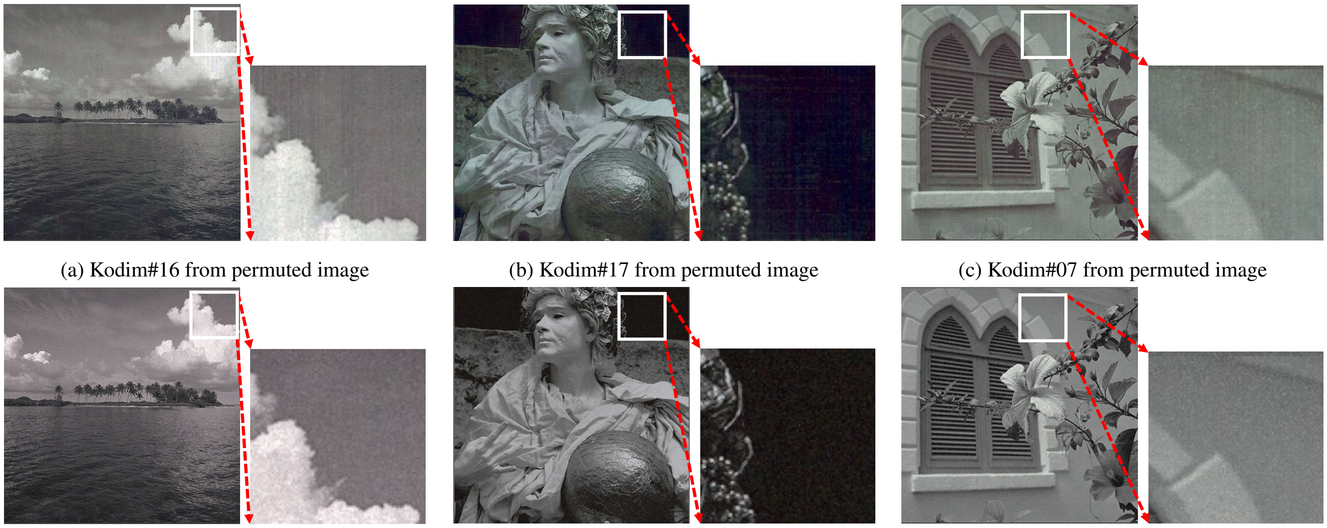
(i) Kodak#21

(j) Kodak#22

(k) Kodak#23

(l) Kodak#24

Figure S12. **PSNR Curves of Kodak images #13–#24.** We report the PSNR curves of Kodak images. The RPP images quickly surge to PSNR 50dB.



(a) Kodim#16 from permuted image

(b) Kodim#17 from permuted image

(c) Kodim#07 from permuted image

(d) Kodim#16 from permuted image

(e) Kodim#17 from permuted image

(f) Kodim#07 from permuted image

Figure S13. **Example reconstructions at a target PSNR value of 30dB (Instant-NGP).** We report the reconstructed images with Instant-NGP at PSNR 30dB. The original images have subtle axis-aligned artifacts.

S7. Reconstructed images in Instant-NGP

We present an extended analysis of the reconstructed images using Instant-NGP at a higher target PSNR value of 30dB. While the increase in PSNR typically correlates with enhanced image fidelity, our observations reveal a subtle, yet noteworthy, persistence of artifacts. These artifacts, manifesting as horizontal and vertical lines, are similar in nature to those observed at the lower PSNR of 20dB Fig. 8, albeit less pronounced. Figure S13 illustrates these findings, showcasing the reconstructed images at PSNR 30dB. We hypothesize that these artifacts are intrinsically linked to the spatial grid encoding of the Instant-NGP model, a pattern consistent with our earlier observations at the lower PSNR threshold.

S8. Additional loss landscapes

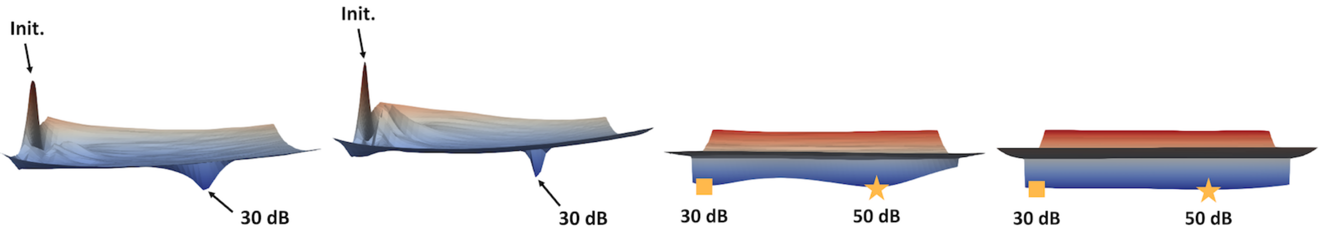
S8.1. Landscapes on other images

We show the loss landscapes for the first two images from Kodak, DIV2K, and CLIC. As shown in Figs. 5 and 6, we select a direction vector between two parameters and, the other direction randomly. In Figs. S14 to S19, (a) and (b) illustrate the loss landscapes during the early phase (i.e. initial point to 30dB), while (c) and (d) correspond to the late phase (i.e. 30dB to 50dB). In the early phase, as illustrated in the most of the figures, the loss landscape of the original image shows a smoother and more navigable trajectory from the initial point to a PSNR of 30dB than the RPP. We also typically observe a ”linear expressway’ in all the loss landscapes of the RPP images, which is detailed in our main paper.

S8.2. Projections on different direction vectors

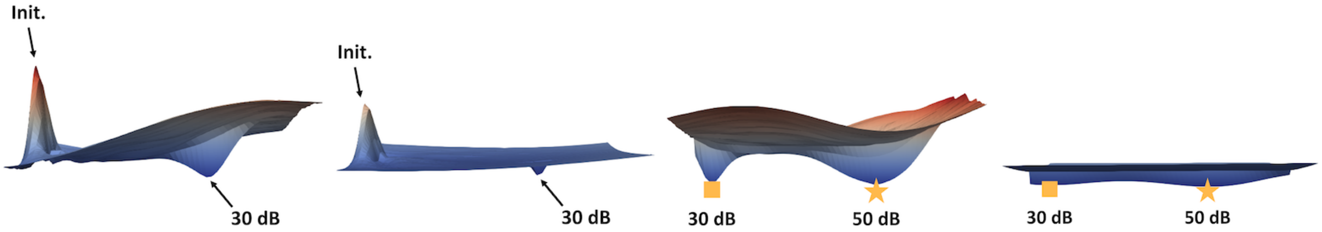
Figure S20 shows the loss landscapes for the Kodak#08 image, which is already used in Figs. 5 and 6. We keep the direction between two parameters, yet in this instance, another direction is chosen differently at random. Despite the difference in random directions, the overall shapes of the loss landscapes exhibit a similarity due to the insignificance of random vectors in a high-dimensional space.

Furthermore, we provide the loss landscapes again for the Kodak#08 image, opting not to select the direction randomly this time. Instead, we use the top-1 eigenvector, corresponding to the largest eigenvalue of the Hessian matrix of the loss function, rather than the random vector. Figures S21 and S22 presents these loss landscapes, offering both elevated and side views as in Figs. 5 and 6, respectively.



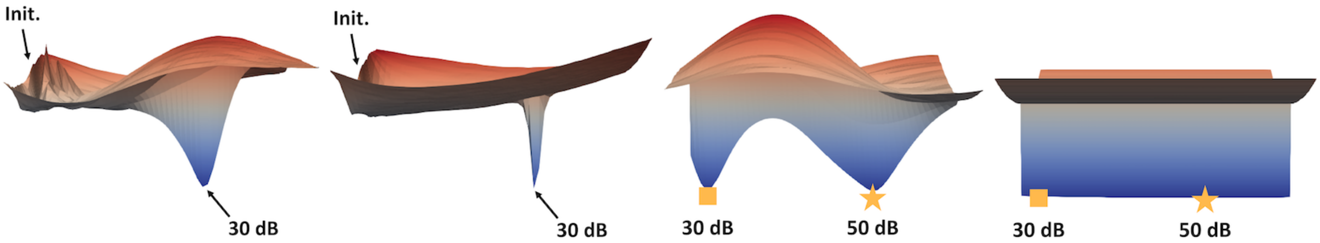
(a) original; init. to 30dB (b) RPP; init. to 30dB (c) original; 30dB to 50dB (d) RPP ; 30dB to 50dB

Figure S14. **SIREN loss landscape: Kodak#01**



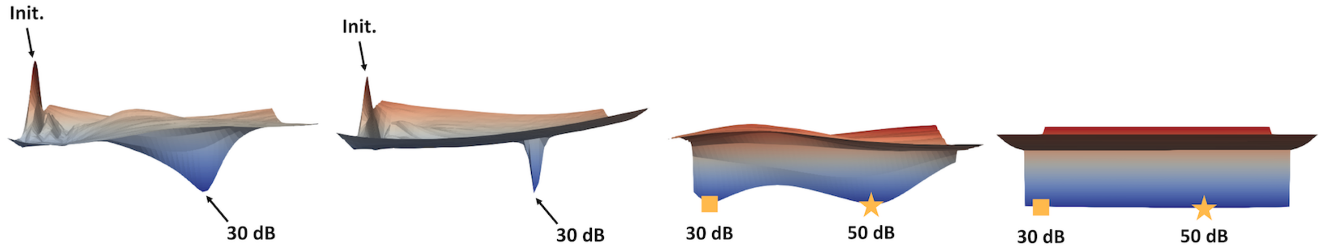
(a) original; init. to 30dB (b) RPP; init. to 30dB (c) original; 30dB to 50dB (d) RPP ; 30dB to 50dB

Figure S15. **SIREN loss landscape: Kodak#02**



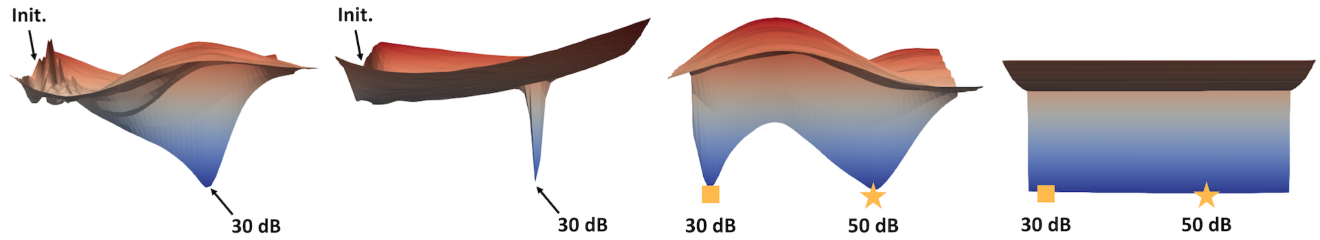
(a) original; init. to 30dB (b) RPP; init. to 30dB (c) original; 30dB to 50dB (d) RPP ; 30dB to 50dB

Figure S16. **SIREN loss landscape: DIV2K#801**



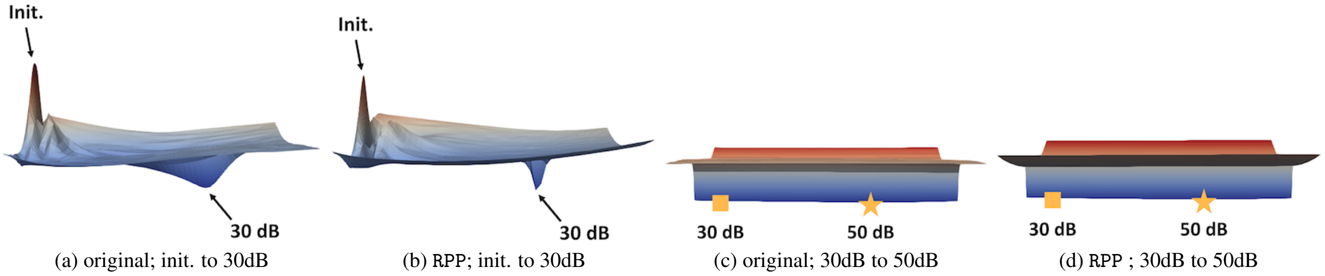
(a) original; init. to 30dB (b) RPP; init. to 30dB (c) original; 30dB to 50dB (d) RPP ; 30dB to 50dB

Figure S17. **SIREN loss landscape: DIV2K#802**



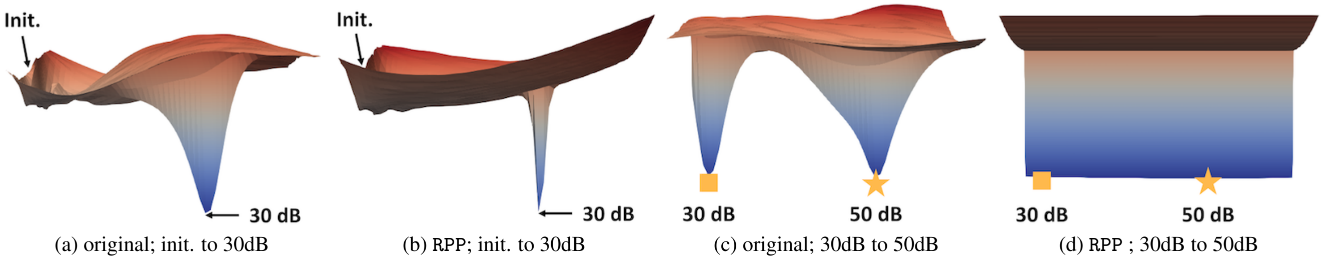
(a) original; init. to 30dB (b) RPP; init. to 30dB (c) original; 30dB to 50dB (d) RPP ; 30dB to 50dB

Figure S18. **SIREN loss landscape: CLIC#01**



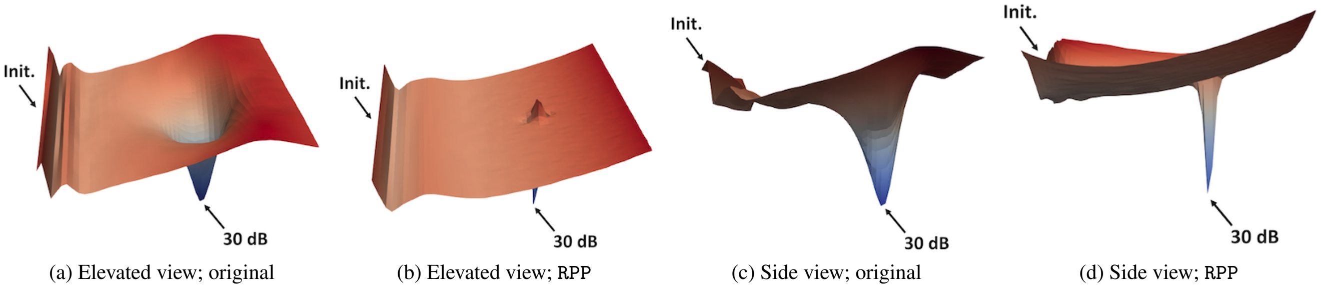
(a) original; init. to 30dB (b) RPP; init. to 30dB (c) original; 30dB to 50dB (d) RPP ; 30dB to 50dB

Figure S19. **SIREN loss landscape: CLIC#02** In this case, the original image is quickly reaches PSNR 50dB (557 steps) than the RPP image (1610 steps)



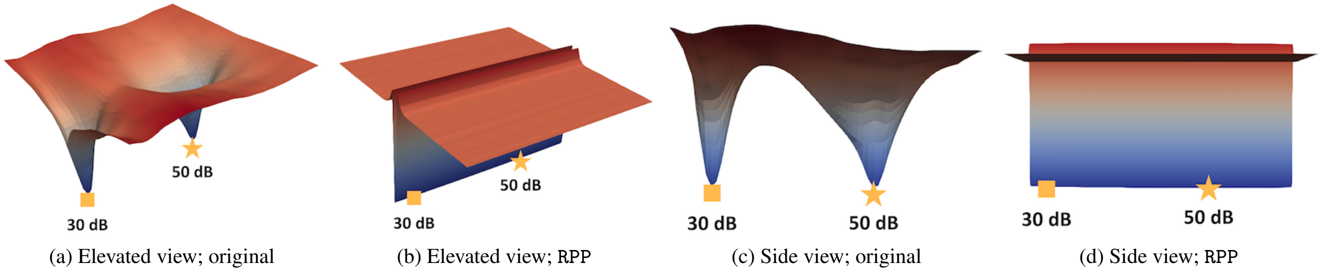
(a) original; init. to 30dB (b) RPP; init. to 30dB (c) original; 30dB to 50dB (d) RPP ; 30dB to 50dB

Figure S20. **SIREN loss landscape with different random direction: Kodak#08**



(a) Elevated view; original (b) Elevated view; RPP (c) Side view; original (d) Side view; RPP

Figure S21. **SIREN loss landscape with eigenvector direction, corresponding to the largest eigenvalue: Kodak#08 (initial to 30dB)**



(a) Elevated view; original (b) Elevated view; RPP (c) Side view; original (d) Side view; RPP

Figure S22. **SIREN loss landscape with eigenvector direction, corresponding to the largest eigenvalue: Kodak#08 (30dB to 50dB)**