In Search of a Data Transformation that Accelerates Neural Field Training

Junwon Seo[1] **Sangyoon Lee**[1] **Jaeho Lee**[1]
[1]POSTECH
`{junwon.seo,lyunm1206,jaeho.lee}@postech.ac.kr`

Abstract

We investigate accelerating the training of neural fields, specialized networks representing individual data points, by fitting transformed data versions; one can recover the original signal by inverting back the signal represented by the trained neural field. We empirically find that very simple data transformations, such as color inversion or random pixel shuffling, can substantially speed up or slow down the training. In particular, to our surprise, we observe that an image with randomly shuffled pixels can be fit much faster, despite having a very large frequency.

1 Introduction

Neural field is a form of data representation that parameterizes each signal as a neural network that maps spatiotemporal coordinates to the signal value [1], useful for high-detail representation of various modalities like images [2], videos [3], 3D scenes [4], and spherical data [5]. Neural fields' widespread use is limited by high training costs, requiring numerous SGD iterations for *each* data representation. For instance, NeRF requires at least 12 hours of training time on GPUs to represent a single 3D scene [4]. Such computation time for data representation hinder their practical application.

The prevailing approach to reduce training time by incorporating useful *prior* (or inductive bias). Many prior works view the implicit bias of SGD as a core cause of the long training time of neural fields. In particular, the spectral bias of standard neural network training is known to impede the fitting of high-frequency signal details. [6]. Such works propose various algorithms to mitigate this SGD bias by including network components like Fourier features [4], sinusoidal activations [7], spatial encoding [8, 9], or using *initial parameters* meta-learned from a large dataset [10, 11].

In this paper, we approach the problem from a different angle, by asking the following question:

*"Can we **exploit** the implicit bias of SGD, instead of fighting it?"*

Precisely, we explore if *transforming* the signal can make SGD bias favorable for fitting the transformed signal. Should an efficiently invertible transformation exist, it may enable reduced training costs by fitting the transformed signal instead. As a first step to answer this question, we compare the training costs (in terms of SGD steps) for fitting both original and transformed signals to a specific fidelity, using an optimal learning rate. Here, we focus on a relatively simple case of image regression, using a recently proposed neural field architecture with multi-resolution hash encoding [9].

Our experiments reveal that simple transformations dramatically affect neural fields' training speed:

- *Random permutation:* We find that random pixel permutations in images lead to fit faster. This challenges the simple notion of 'low-frequency images being easier to fit' of spectral bias and indicates a need for deeper insights into how frequency interacts with architecture and batch size.
- *Color inversion:* Color-inverted images require $\sim 7\times$ more SGD steps on average to be trained; we attribute this phenomenon to the neural networks' tendency to fit low-magnitude signals better.

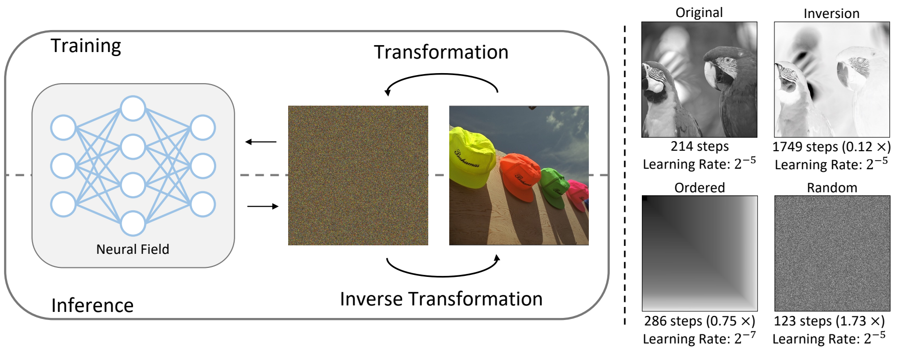

Figure 1: **Overall pipeline.** fits a neural field to a data transformed via an invertible mapping, enabling recovery of the original data from the neural field output (left). Careful selection of this transformation can greatly reduce the SGD steps needed for a desired approximation quality (right).

2 Framework

The general framework of finding data transformations that accelerate neural field training can be formalized as an optimization problem as follows.

Suppose that we are given some *signal* $\mathbf{x} \in \mathcal{X}$ that we want to fit with a neural field. Here, the *signal space* $\mathcal{X}$ may be the space of all signals that has some data type (e.g. 256×256 RGB images could be described as a subset of $\mathbb{R}^{256\times256\times3}$). We consider transforming this signal using some *transformation* $T : \mathcal{X} \to \mathcal{X}$, $T \in \mathcal{T}$, where the *transformation space* $\mathcal{T}$ is the set of all invertible transformations. We aim to find a nice transformation T that reduces the training cost for the transformed signal $T(\mathbf{x})$.

To formalize this goal, we define the *training cost* function $\mathsf{cost} : \mathcal{X} \times \mathcal{T} \to \mathbb{R}$. The training cost $\mathsf{cost}(\mathbf{x}, T)$ measures the training cost required to train a neural field for $T(\mathbf{x})$, whenever its outcome is inverted back via T^{-1}, approximates the original signal $\mathbf{x}$ with some desired level of precision. Note that this cost need not be identical to the training cost of fitting $T(\mathbf{x})$ with the same precision, as the transformation may involve scaling of the signal. The Cost depends on architectures, hardware, and batch size—using a larger batch size can decrease convergence steps [12] but requires more memory.

Given these, we want to solve the optimization problem

$$\text{minimize} \quad \mathsf{cost}(\mathbf{x}, T), \qquad \text{subject to} \quad T \in \mathcal{T}^*, \tag{1}$$

where $\mathcal{T}^* \subseteq \mathcal{T}$ is a subset of the transformation space that satisfies some desired properties, such as

- *Efficiently invertible*: The inverse transformation T^{-1} needs to be efficiently computable (e.g. a linear operation). Ideally, it should be achievable by directly modifying the trained parameters.
- *Retains interpolatability*: For neural field applications like super-resolution [2], we need T to allow principled sampling of interpolated coordinates from the approximated transformed signal.

Problem 1 can be solved either for a datum $\mathbf{x}$ or a distribution of data $P \in \mathcal{P}(\mathcal{X})$. In the latter case, the goal is to *learn* a transformation that minimizes the expected training cost over the distribution.

3 Methodology

In this paper, as a preliminary step to solve the problem 1, we focus on the *proof-of-concept* that there exists some transformation such that the training cost reduces over the vanilla neural field training. More specifically, we show that there exists some choice of T such that

$$\mathsf{cost}(\mathbf{x}, T) < \mathsf{cost}(\mathbf{x}, \mathrm{Id}) \tag{2}$$

under a specific choice of the neural field architecture, signal and the cost function. More specifically, we focus on the task of 2D image regression, and measure the *number of SGD steps* required to reach training PSNR 50, given the optimal learning rate (tuned from $\{2^{-4}, \ldots, 2^{-13}\}$); the number of SGD steps is an informative indicator of the total FLOPs and runtime for any fixed choice of hardware, model architecture, and data. Other experimental configurations are as follows.

Table 1: **Average speedup on Instant-NGP.** We measure how each image transformation affects the number of SGD steps until convergence on Kodak images, and report the average speedup ratio against the original image. The speedups are colored in green, and the slowdowns are colored in red.

	Color Inversion	Random Permutation	Ordered Permutation
Average Speedup	0.123×	1.360×	0.765×

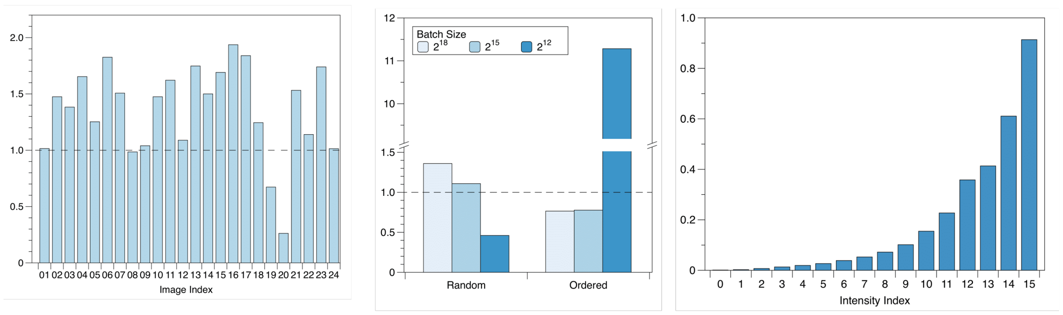

Figure 2: **Per-image speedup from random permutation (left).** We measure the speedup ratio from random pixel permutation in images, observing over +50% speedup in nearly half of the cases. **The effect of batch size on permutation (mid).** Randomly pixel-permuted images show improved efficiency with larger batch sizes, while ordered pixel-permuted images fit better with smaller batches. **Intensity vs. Loss (right).** We report the average squared losses of the pixels that have similar intensities. Lower index correspond to lower intensities.

Data. 512×512 grayscale images, center-cropped from Kodak dataset [13] with intensities in $[0, 1]$.

Model. We mainly use Instant-NGP architecture with multi-resolution hash encoding [9]. We include SIREN [7] and ReLU networks with sinusoidal positional encodings [4] in certain experiments.

Training. We train the neural field using the mini-batch gradient descent. We use full-batch as a default (i.e., 2^{18}), as this typically minimizes the number of required iterations. We also consider batch sizes of $\{2^{12}, 2^{15}\}$ to study the impact of data transformations on acceleration. Data batches are sampled directly from coordinate points, not from an interpolated target image (used, e.g., in Instant-NGP [9]).

Data Transformations. We consider total three transformations.

- **Color inversion**: We invert the target image, i.e., perform $z \mapsto 1 - z$ on the pixel-wise intensity values. The inverse transformation is identical to the transformation itself.
- **Random permutation**: We shuffle pixel positions, keeping their intensity values, and use the permutation matrix's transpose for inverse transformation, resulting in a high-frequency image.
- **Ordered permutation**: We arrange pixels in ascending intensity order, starting at the top-left corner with a 1×1 square. Expanding to a 2×2 square, we fill the ⅃-shaped region clockwise with the next three pixels, then counterclockwise for subsequent ⅃-shaped areas in larger squares. This process continues until the full image is filled, creating a low-frequency image.

4 Observations

Table 1 shows that data transformations notably impact SGD efficiency, causing an $\sim 88\%$ slowdown to a 36% speedup. We identify two key observations for further experiments and propose initial hypotheses for each.

4.1 Randomly pixel-permuted images can be fit faster

We found that random pixel permutations often accelerate image training, as shown in Figure 2's left side. Out of 24 images, 21 show speedups, with up to +93.7% (image#16), though some experience significant slowdowns, like −32.7% on image#19 and −73.8% on image#20. Conversely, ordered permutation generally slows training by −23.5%. This challenges assumptions about frequency in

Table 2: **Random permutation on various architectures.** We measure the average speedup from random permutation on various architectures, including the models that use sinusoidal encodings.

	Instant-NGP	SIREN	ReLU+P.E. (m=10)	ReLU+P.E. (m=15)
Average Speedup	1.360×	1.265×	0.825×	1.099×

Table 3: **Fitting darker/brighter images.** We measure how each image darkening or brightening transformations affect the training time. Note that, unlike prior tables and figures, we measure the number of SGD steps until achieving PSNR 50 on the *transformed signal*, not on the original signal.

	Halving	Gamma Correction			
		1/4	1/2	3/2	2
Average Speedup	1.125×	2.178×	1.438×	0.766×	0.489×

training: randomly permuted images (DCT[1] intensity 35.67) train faster than low-frequency ones (intensity 3.58). Ordered permutations result in low-frequency images with an intensity of just 0.03. This defies the assumption that low-frequency images train quicker. Additional experiments were performed to elucidate the reasons for this unexpected outcome.

- **Batch size:** We found that the speed of fitting pixel-permuted images is largely dependent on batch size (mid of Figure 2). Larger batch sizes favor random permutations, while smaller ones favor ordered permutations, which are on average 11× faster. We hypothesize that the high SGD noise in small batch size hinders the model's ability to fit high-frequency signals.
- **Architecture.** We discovered that the acceleration from random permutations is largely architecture-dependent (Table 2). Models with greater expressive capacity perform better with random pixels, especially evident in ReLU networks with positional encodings [4]. Model with more sinusoidal bases ($m = 10, 15$) gain from random pixels, while other model with fewer bases are hindered.

From these observations, We hypothesize that random permutations speed up training by enabling better utilization of the model's capacity for high-frequency components, typically underutilized. With small batch sizes, the SGD noise becomes too large so that it reduces the capacity of the models to fit high-frequency signals; the ability to fit low-frequency signals (for some reason) tends to be more robust to such noise.

4.2 Inverted images fit much slower

Our findings show that original images fit significantly faster than inverted ones, requiring only about 12.3% of the SGD steps. The exception is Kodak image#20, which requires a similar number of steps for both. We suggest this could be due to a *magnitude bias* where pixels with higher intensity are harder to fit. To back-up this point, we conduct follow-up analyses.

Table 3 shows that **darkening/brightening transformations**, via intensity halving (i.e., $z \mapsto z/2$) or gamma correction (i.e., $z \mapsto z^{1/\gamma}$), impact SGD fitting; darker images fit more easily than brighter ones. Right of Figure 2 shows our analysis of the **intensity-to-loss relationship**, revealing higher losses for higher intensity pixels, suggesting neural fields' increased struggle with larger-magnitude functions.

5 Discussion, Limitations and Future Direction

In our study, we show that certain data transformations, like pixel permutations and color inversion, significantly affect neural field training costs. These effects are linked to a refined understanding of spectral and magnitude biases in neural network training. However, the practicality of our findings is currently limited; for example, random permutation hampers interpolation. And our work's scope is mostly confined to image regression using a specific architecture. As a future work, we aim to distill the core principles, namely the spectral and magnitude bias, from our empirical observations and use them as a main toolbox to construct a nice family of data transformation which presumably contains many useful options for solving the problem 1.

[1]We measure this as follows: we divide each image into 8×8 patches and compute the DCT coefficients. We compute the sum of squared coefficients, except for the low-frequency region (i.e., the upper-left quarter).

Acknowledgments

This work was supported by the National Research Foundation of Korea(NRF) grant funded by the Korea government(MSIT) (No.RS-2023-00213710, Neural Network Optimization with Minimal Optimization Costs) and by the Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No.2022-0-00713, Meta-learning applicable to real-world problems).

References

[1] Yiheng Xie, Towaki Takikawa, Shunsuke Saito, Or Litany, Shiqin Yan, Numair Khan, Federico Tombari, James Tompkin, Vincent Sitzmann, and Srinath Sridhar. Neural fields in visual computing and beyond. *Computer Graphics Forum*, 2022.

[2] Yinbo Chen, Sifei Liu, and Xiaolong Wang. Learning continuous image representation with local implicit image function. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021.

[3] Subin Kim, Sihyun Yu, Jaeho Lee, and Jinwoo Shin. Scalable neural video representations with learnable positional features. In *Advances in Neural Information Processing Systems*, 2022.

[4] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. NeRF: Representing scenes as neural radiance fields for view synthesis. In *European Conference on Computer Vision*, 2020.

[5] Daniele Grattarola and Pierre Vandergheynst. Generalised implicit neural representations. *Advances in Neural Information Processing Systems*, 2022.

[6] Nasim Rahaman, Aristide Baratin, Devansh Arpit, Felix Draxler, Min Lin, Fred Hamprecht, Yoshua Bengio, and Aaron Courville. On the spectral bias of neural networks. In *Proceedings of the International Conference on Machine Learning*, 2019.

[7] Vincent Sitzmann, Julien N.P. Martel, Alexander W. Bergman, David B. Lindell, and Gordon Wetzstein. Implicit neural representations with periodic activation functions. In *Advances in Neural Information Processing Systems*, 2020.

[8] Julien N. P. Martel, David B. Lindell, Connor Z. Lin, Eric R. Chan, Marco Monteiro, and Gordon Wetzstein. ACORN: Adaptive coordinate networks for neural scene representation. *ACM Transactions on Graphics*, 40(4), 2021.

[9] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM Transactions on Graphics*, 41(4):102:1–102:15, 2022.

[10] Vincent Sitzmann, Eric Chan, Richard Tucker, Noah Snavely, and Gordon Wetzstein. MetaSDF: Meta-learning signed distance functions. In *Advances in Neural Information Processing Systems*, 2020.

[11] Matthew Tancik, Ben Mildenhall, Terrance Wang, Divi Schmidt, Pratul P. Srinivasan, Jonathan T. Barron, and Ren Ng. Learned initializations for optimizing coordinate-based neural representations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021.

[12] Christopher J. Shallue, Jaehoon Lee, Joseph Antognini, Jascha Sohl-Dickstein, Roy Frostig, and George E. Dahl. Measuring the effects of data parallelism on neural network training. *Journal of Machine Learning Research*, 20(112):1–49, 2019.

[13] E. Kodak. Kodak dataset, 1999.